­МИНИСТЕРСТВО НАУКИ И ВЫСШЕГООБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**ЛАБОРАТОРНАЯ РАБОТА**

на тему:

**«Битовые поля и множества»**

**Выполнил(а):** студент(ка) группы 3822Б1ФИ2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Холин К.И./

Подпись

**Проверил:** к.т.н, доцент каф. ВВиСП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Кустикова В.Д./

Подпись

Нижний Новгород  
2023

**Содержание**

[Введение 3](#_Toc147915966)

[1 Постановка задачи 4](#_Toc147915967)

[2 Руководство пользователя 5](#_Toc147915968)

[2.1 Приложение для демонстрации работы битовых полей 5](#_Toc147915969)

[2.2 Приложение для демонстрации работы множеств 5](#_Toc147915970)

[2.3 «Решето Эратосфено» 5](#_Toc147915971)

[3 Руководство программиста 6](#_Toc147915972)

[3.1 Описание алгоритмов 6](#_Toc147915973)

[3.1.1 Битовые поля 6](#_Toc147915974)

[3.1.2 Множества 6](#_Toc147915975)

[3.1.3 «Решето Эратосфена» 6](#_Toc147915976)

[3.2 Описание программной реализации 6](#_Toc147915977)

[3.2.1 Описание класса TBitField 6](#_Toc147915978)

[3.2.2 Описание класса TSet 7](#_Toc147915979)

[Заключение 8](#_Toc147915980)

[Литература 9](#_Toc147915981)

[Приложения 10](#_Toc147915982)

[Приложение А. Реализация класса TBitField 10](#_Toc147915983)

[Приложение Б. Реализация класса TSet 10](#_Toc147915984)

# Введение

В C++ иногда возникают такие ситуации, когда информацию об объекте достаточно хранить в формате состояний (статусов), представляющих из себя 0 и 1. На этом основывается представление множества в виде характеристического массива. Это оптимальный вариант, поскольку формально обеспечивает хранение признака наличия или отсутствия элемента универса во множестве, а не прямое хранение элементов. Обращение к определённому биту позволяет нам узнать его состояние для выполнения конкретной задачи. Битовые поля в этом случае играют важную роль.

# Постановка задачи

Цель – реализовать классы для представления битовых полей TBitField и множеств TSet.

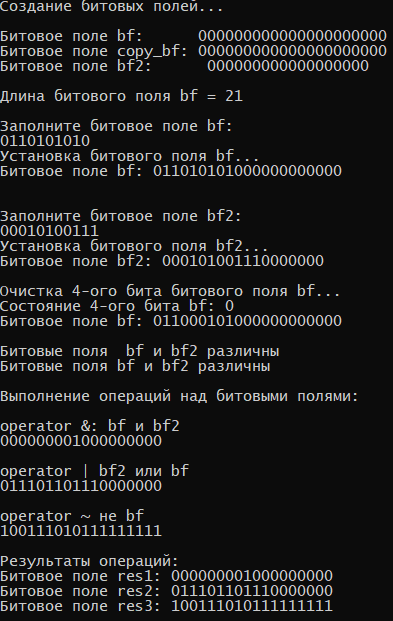
Задачи:

1. Класс для работы с множествами должен поддерживать эффективное хранение данных.
2. Написать следующие операции для работы с битовыми полями: установить бит в 1,установить бит в 0,получить значение бита,сравнить два битовых поля,сложить и инвертировать,вывести битовое поле требуемого формата и ввести битовое поле.
3. Добавить вспомогательные операции получения бита,маски бита,длины битового поля.
4. Написать следующие операции для работы с множествами: вставка элемента,удаление,проверка наличия,сравнение множеств,объединение множеств,пересечение,разность,копирование,вычисление мощности множества,вывод элементов множества требуемого формата и ввод.
5. Добавить вспомогательные операции для получения мощности множества.

# Руководство пользователя

## Приложение для демонстрации работы битовых полей

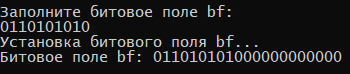
1. Запустите приложение с названием sample\_TBitField.exe. В результате появится окно, показанное ниже ().



1. Основное окно программы
2. На первом шаге создаются 3 битовых поля ()



1. Создание битовых полей.
2. На следующем шаге выполняется установка битового поля bf и выводится его длина (рис.3).



1. Установка битового поля bf с выводом длины
2. Далее выполняется установка битового поля bf2(рис.4)

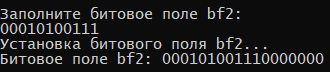


Рис.4 Установка битового поля bf2

1. На 5 шаге удаляется бит с номером 4 из битового поля bf(рис.5)



Рис.5 Удаление 4-го бита битового поля bf2

1. В первой строке проверяется операция равенства битовых полей bf и bf2,а во второй- операция неравенства(рис.6)



Рис.6 Сравнение битовых полей

1. На данном этапе выполняются различные операции с битовыми полями(рис.7)

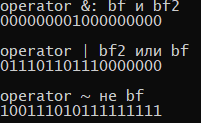


Рис.7 Операции над битовыми полями

1. На завершающем шаге выводятся результаты вычислений(рис.8)

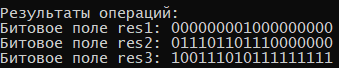


Рис.8 Результаты вычислений

## Приложение для демонстрации работы множеств

1. Запустите приложение с названием sample\_tset.exe. В результате появится окно, показанное ниже ().

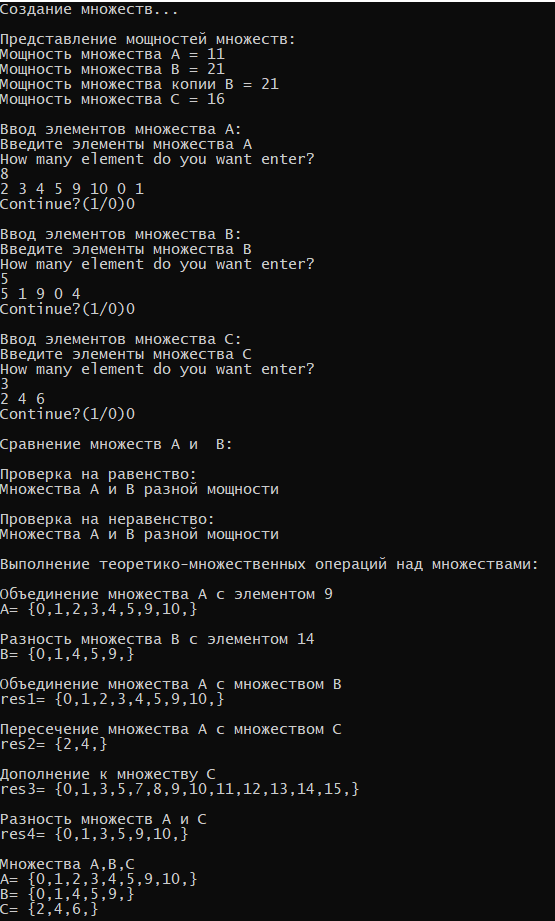


Рис.1 Окно основной программы

1. По началу множества пустые,так как в них не содержится элементов. Далее представлены мощности множеств(рис.2)



Рис.2 Мощности множеств A,B,C

1. На третьем этапе представлен процесс заполнения множеств элементами, введёными с клавиатуры(рис.3)

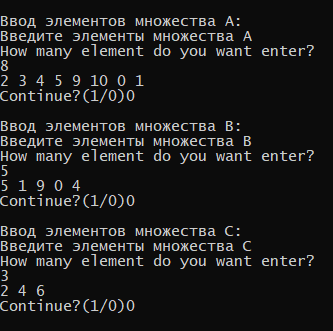


Рис.3 Заполнение множеств A,B,C

1. В этом случае сравниваются два множества на равенство и неравенство(рис.4)

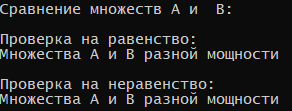


Рис.4 Сравнение множеств A и B

1. На рис.5 приведены основные операции с множествами(рис.5)

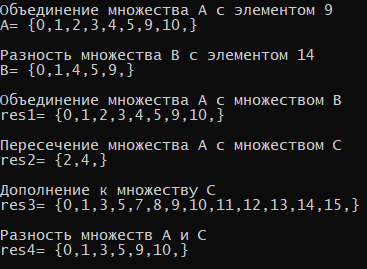


Рис.5 Основные операции с множествами A,B,C

1. В завершение были выведены множества A,B,C, которые принимали участие в программе(рис.6)

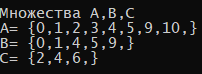


Рис.6 Вывод множеств A,B,C

## «Решето Эратосфена»

1. Откройте приложение sample\_primenumbers.exe.В результате появится окно ниже (рис.1).

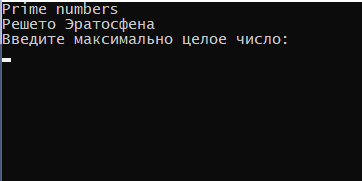
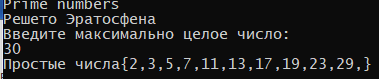


Рис.1 Окно основной программы

1. Вам будет необходимо ввести число ,до которого будут выведены все простые числа на экран. Для примера введём число 30 и посмотрим на результат(рис.2)



1. Рис.2 Все простые числа от 2 до 30

# Руководство программиста

## Описание алгоритмов

### Битовые поля

Битовые поля представляют из себя последовательность нулей и единиц. Элемент битового поля может находиться в двух состояниях: 1 и 0. 1- элемент содержится в множестве, а 0 – элемент не содержится в множестве. Данный алгоритм позволяет реализовать интерфейс для работы с множествами.

Пусть дано множество A:

A = { 0, 2, 3 , 4, 6 }

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| bits | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |

Битовое поле:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |

Описания методов:

Конструктор инициализатор

Входные параметры: len – длина битового поля.

Выходные параметры: отсутствуют.

Алгоритм:

1. Вычисляется количество элементов памяти для хранения битового поля.
2. Выделяется память под массив для хранения битового поля размера MemLen.
3. Функция memset инициализирует память нулевыми значениями.

Конструктор копирования.

Входные данные: константная ссылка на объект типа TBitField.

Выходные данные: отсутствуют.

Алгоритм:

Копирует значения полей переданного объекта в поля текущего объекта.

Деструктор.

Входные данные: отсутствуют.

Выходные данные: отсутствуют

Алгоритм:

1. Освобождает выделенную память из-под массива pMem.
2. Устанавливает значения полей объекта в 0.

.

GetMemMask

Входные данные: n – номер бита

Выходные данные: число типа TELEM,где TELEM – тип элементов массива pMem.

Алгоритм:

Выполняет побитовый сдвиг единицы влево на (n & (BitsInMem-1) бит, где BitsInMem – это количество битов одной единице памяти.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | n | … | … | 0 | 0 | 0 |

GetLength

Входные данные: отсутствуют.

Выходные данные: BitLen – длина битового поля.

Алгоритм:

Возвращает длину битового поля.

SetBit

Входные данные: n – номера бита.

Выходные данные: отсутствуют.

Алгоритм:

1. Проверка номера бита на границы битового поля.

2. Выполняем операцию побитового сложения с элементом и битовой маской по номеру бита n. Используются методы GetMemIndex и GetMemMask соответственно.

3. Результат сложения записывается по номеру элемента в памяти. Используется метод GetMemIndex.

ClrBit

Выходные данные: n – номер бита.

Входные данные: отсутствуют.

Алгоритм:

1. Проверка номера бита на границы битового поля.

2. Выполняем операцию побитового умножения с элементом и битовой маской(битовая маска инвертируется с помощью операции ~) Используются методы GetMemIndex и GetMemMask соответственно.

3. Результат умножения записывается по номеру элемента в памяти.

GetBit

Входные данные: n – номер бита

Выходные данные: значение бита с номером n

Алгоритм:

Возвращает значение бита с номером n.

Operator!=

Входные данные: константная ссылка на тип TBitField

Выходные данные: число: 0 или 1

Алгоритм:

1. Выполняет побитовое сравнение значений битовых полей.
2. Возвращает 0 – битовые поля не не равны, или 1 – битовые поля не равны

Operator==

Входные данные: константная ссылка на тип TBitField

Выходные данные: число 0 или 1

Алгоритм:

1.Выполняет побитовое сравнение значений битовых полей.

2 .Возвращает 0 – битовые поля не равны, или 1 – битовые поля равны

Operator=

Входные данные: константная ссылка на объект типа TBitField

Выходные данные: \*this – ссылка на объект себя

Алгоритм:

1. Проверка на самоприсваивание.
2. Проверка на равенство длин битовых полей. Иначе память перевыделяется.
3. Копирование значений элементов переданного объекта в текущий объект.
4. Возвращается \*this.

Operator&

Входные данные: константная ссылка на объект типа TBitField

Выходные данные: объект типа TBitField

Алгоритм:

1.Создаётся копия текущего объекта

2. Побитовое умножение битовых полей. Если оба значения бита равны 1,результат равен 1 . Иначе если хотя бы 1 ноль,то результирующий бит равен 0.

3. Возвращается объект типа TBitField как результат побитового умножения.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BF1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| BF2 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| BF1&BF2 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

Operator~

Входные данные: отсутствуют

Выходные данные: объект типа TBitField

Алгоритм:

1. Проверка на равенство размером. Иначе создаётся временный объект, выделяется память под нужный размер, значения старой памяти копируются в новую и старая память удаляется.
2. Указатель на старую память получает значение указателя на выделенную память и временный объект удаляется.
3. Выполняется побитовое сложение. Если при сложении значений битовых полей оба бита равны 0,то результирующее значение 0. Если хотя бы 1 единица,то результирующее значение 1.
4. Возвращается объект типа TBitField как результат инвертации.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BF1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| ~BF1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

Operator|

Входные параметры: константная ссылка на объект типа TBitField

Выходные параметры: объект типа TBitField

Алгоритм:

1. Проверка на равенство размеров. Иначе создаётся временный объект, выделяется память под нужный размер, значения старой памяти копируются в новую и старая память удаляется.
2. Указатель на старую память получает значение указателя на выделенную память и временный объект удаляется.
3. Выполняется побитовое сложение. Если при сложении значений битовых полей оба бита равны 0,то результирующее значение 0. Если хотя бы 1 единица,то результирующее значение 1.
4. Возвращается объект типа TBitField как результат побитового сложения.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BF1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| BF2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| BF1|BF2 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

Operator>>

Входные параметры: istr - ссылка на стандартный поток ввода, неконстантная ссылка на объект типа TBitField.

Выходные параметры: istr – ссылка на стандартный поток ввода.

1. Вводится посл-ть из 0 и 1
2. Если значение 0,то вызывается метод ClrBit и соответствующее значение бита битового поля устанавливается в 0. В противном случае вызывается метод SetBit.
3. Возвращается ссылка на стандартный поток istr.

Operator<<

Входные данные: ostr – ссылка на стандартный поток вывода,константная ссылка на объект типа TBitField.

Выходные данные:ostr - ссылка на стандартный поток вывода

Алгоритм:

1. Открывается фигурная скобка {

2. Выполняется проверка на принадлежность i-того элемента множества. Если i-тый элемент принадлежит множеству,то выводится на консоль с помощью метода GetBit. Иначе счётчик увечивается на 1 и итерация повторяется

3. Закрывается фигурная скобка }.

4. Возвращается ссылка на стандартный поток вывода.

### Множества

Множества по идее наследуются от класса TBitField. Множество – это класс TSet,реализованный на основе класса TBitField. Работа TSet заключается в том,что он использует класс TBitField как инструмент для создания множеств и осуществления теоретико-множественных операций

Описания методов:

Конструктор инициализатор.

Входные параметры: mp- максимальная мощность множества

Выходные параметры: отсутствуют

Алгоритм:

1. Поле MaxPower инициализируется значением mp.

2. Присваивание полю BitField значения bf..

Конструктор копирования.

Входные значения: константная ссылка на объект типа TSet

Выходные значения: отсутствуют

Алгоритм:

Выполняется копирование значений полей переданного объекта в текущий объект.

Конструктор преобразования:

Входные данные: константная ссылка на объект типа TBitField

Выходные данные: отсутствуют

Алгоритм:

1. MaxPower инциализируется значением len.

2. Присваивание TBitField и bf.

Operator TBitField()

Входные параметры: отсутствуют

Выходные параметры: объект типа TBitField

Алгоритм:

1.Создаётся новый объект типа TBitField с вызовом конструктора копирования.

2. Возвращается новый объект типа TBitField.

GetMaxPower

Входные данные: отсутствуют

Выходные данные: MaxPower- максимальная мощность множества.

Алгоритм:

Возвращается максимальная мощность множества.

InsElem

Входные данные: elem- элемент включения во множество

Выходные данные: отсутствуют

Алгоритм:

Вызывает метод Setbit с переданным параметром elem.

DelElem

Входные параметры: elem – элемент для исключения из множества

Выходные параметры: отсутствуют

Алгоритм:

Вызывается метод ClrBit с переданным параметром elem.

IsMember

Входные параметры: elem- элемент для проверки на принадлежность множеству

Выходные параметры: число: 0 или 1

Алгоритм:

1. Вызывает метод GetBit с переданным параметром elem.
2. Возвращает результат вызова метода GetBit. 1 – элемент принадлежит множеству,0 – не принадлежит.

Operator==

Входные параметры: константная ссылка на объект типа TSet

Выходные параметры: число: 0 или 1

Алгоритм:

1. Выполняется сравнение битовых полей двух множеств.
2. Возвращается 0 – множества равны,0 – не равны

Operator!=

Входные параметры: константная ссылка на объект типа TSet

Выходные параметры: число: 0 или 1

Алгоритм:

1.Выполняется сравнение битовых полей двух множеств.

2.Возвращается 0 – множества не не равны. 1 – не равны

Operator=

Входные параметры: константная ссылка на объект типа TSet

Выходные параметры: \*this – ссылка на объект себя.

Алгоритм:

1. Проверка на самоприсваивание.
2. Присваивание BitField и bf.
3. Возвращается \*this

Operator+

Входные параметры: elem – элемент для объединения с множеством

Выходные параметры: новый объект TSet с объёдинённым элементом

Алгоритм:

1. Создаётся новый объект TBitField – копия BitField Вызывается метод SetBit с переданным параметром elem.
2. Возвращается конструктор преобразования с результирующим объектом.

A = { 1 , 2 , 3} MaxPower = 6, elem = 5

A` = {1,2,3,5}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | 0 | 1 | 1 | 3 | 0 | 0 |
| A+elem | 0 | 1 | 1 | 1 | 0 | 1 |

Operator-

Входные параметры: elem – элемент для вычитания из множества

Выходные параметры: новый объект TSet с удалённым элементом

Алгоритм:

1.Создаётся новый объект TBitField – копия BitField

2.Вызывается метод ClrBit с переданным параметром elem.

3. Возвращается конструктор преобразования TSet с результирующим объектом.

A = { 0 , 1 , 2,4,5} MaxPower = 6, elem = 5

A` = {0,1,2,4}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | 1 | 1 | 1 | 0 | 1 | 1 |
| A-elem | 1 | 1 | 1 | 0 | 1 | 0 |

Operator+

Входные параметры: константная ссылка на объект типа TSet

Выходные данные: новый объект TSet с результатом объединения множеств.

Алгоритм:

1. Создаётся новый объект TBitField – копия BitField

2. Выполняется побитовое сложение двух битовых полей.

3. Возвращается конструктор преобразования TSet с результирующим объектом.

A= {1,2,4,5},MaxPower = 6

B = {0,3} MaxPower = 6

A|B = {0,1,2,3,4,5}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | 0 | 1 | 1 | 0 | 1 | 1 |
| B | 1 | 0 | 0 | 1 | 0 | 0 |
| A|B | 1 | 1 | 1 | 1 | 1 | 1 |

Operator~

Входные данные: отсутствуют

Выходные данные: новый объект типа TSet

Алгоритм:

1. Создаётся новый объект типа TBitField – копия BitField

2. Применяется операция ~для битового поле.

3. Возвращается конструктор преобразования TSet с результирующим объектом.

A = {1,4,5} MaxPower = 6

~A= { 0,2,3}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | 0 | 1 | 1 | 0 | 1 | 1 |
| ~A | 1 | 0 | 0 | 1 | 0 | 0 |

Operator&

Входные параметры: константная ссылка на объект типа TSet

Выходные параметры: новый объект типа TBitField с объединением множеств.

Алгоритм:

1. Создаётся новый объект типа TBitField – копия BitField.

2. Выполняет побитовое умножение битовых полей.

3. Возвращается конструктор преобразования TSet с результирующим объектом.

A= {1,2,4,5,0},MaxPower = 6

B = {0,3,1} MaxPower = 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | 0 | 1 | 1 | 0 | 1 | 1 |
| B | 1 | 1 | 0 | 1 | 0 | 0 |
| A|B | 0 | 1 | 0 | 0 | 0 | 0 |

Operator-

Входные данные: константная ссылка на объект типа TSet

Выходные параметры: новый объект типа TBitField с исключёнными элементами.

Алгоритм:

1. Создаётся новый объект 1 типа TBitField – копия BitField.

2. Создаётся новый объект 2 типа TBitField – копия переданного объекта.

3. Выполняется побитовое умножение объекта 1 и инвертированного объекта 2.

4. Возвращается конструктор преобразования TSet с результирующим объектом.

A= {1,2,3,5},MaxPower = 6

B = {2,3,4,0} MaxPower = 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | 0 | 1 | 1 | 1 | 0 | 1 |
| B | 1 | 0 | 1 | 1 | 1 | 0 |
| ~B | 0 | 1 | 0 | 0 | 0 | 1 |
| A-B | 0 | 1 | 0 | 0 | 0 | 1 |

Operator>>

Входные данные: : istr - ссылка на стандартный поток ввода, неконстантная ссылка на объект типа TBitField.

Выходные данные: ссылка на стандартный поток ввода

Алгоритм:

1. Вводится кол-во элементов, которое необходимо добавить в множество.
2. Вводятся последовательно некоторые числа(элементы множества)
3. Вызывается метод InsElem.
4. Возвращается ссылка на стандартный поток ввода

Operator<<

Входные данные:

Выходные данные:

Алгоритм:

1. Открывается фигурная скобка {
2. Получаем размер множества.
3. В цикле делаем проверку на принадлежность i-того элемента множеству.
4. Если i-тый элемент принадлежит множеству,то выводится на экран i-тый элемент. Иначе следующая итерация.
5. Закрывается фигурная скобка.
6. Возвращается ссылка на стандартный поток вывода.

### «Решето Эратосфена»

Решето Эратосфена – это алгоритм, позволяющий найти все простые числа до заданного числа n. Суть этого алгоритма заключается в следующем:

1. Выписать подряд все числа от 2 до n
2. Пусть у нас есть переменная p=2 –первое простое число
3. Зачёркиваем все числа,кратные 2p,3p,4p…
4. Находим первое простое число в списке,большее p. Присваиваем его p
5. Повторяем шаги 3 и 4.

Данный алгоритм позволяет легко и быстро найти все простые числа.

## Описание программной реализации

### Описание класса TBitField

class TBitField

{

private:

int BitLen;

TELEM \*pMem;

int MemLen;

// методы реализации

int GetMemIndex(const int n) const;

TELEM GetMemMask (const int n) const;

int BitsInMem = 16;

int shiftsize = 4;

public:

TBitField(int len);

TBitField(const TBitField &bf);

~TBitField();

// доступ к битам

int GetLength(void) const;

void SetBit(const int n);

void ClrBit(const int n);

int GetBit(const int n) const;

// битовые операции

int operator==(const TBitField &bf) const;

int operator!=(const TBitField &bf) const;

const TBitField& operator=(const TBitField &bf);

TBitField operator|(const TBitField &bf);

TBitField operator&(const TBitField &bf);

TBitField operator~(void);

friend istream& operator>>(istream& istr, TBitField& obj);

friend ostream &operator<<(ostream &ostr, const TBitField &bf);

};

Назначение: представление битового поля.

Поля:

BitLen – длина битового поля – максимальное количество битов.

pMem – память для представления битового поля.

MemLen – количество элементов для представления битового поля.

|  |  |  |  |
| --- | --- | --- | --- |
| Методы | | | |
| GetMemIndex  Назначение:  Получение индекса элемента памяти  Входные  Параметры-  n- Номер бита  Выходные параметры  Номер элемента памяти | GetMemMask  Назначение:  Получение битовой маски по номеру бита  Входные  Параметры-  n- Номер бита  Выходные  Параметры:  Битовая маска | GetLength  Назначение:  Получение длины  битового поля  Входные  Параметры:  Отсутствуют  Выходные параметры:  Длина битового поля | SetBit  Назначение:  Установить бит в единицу  Входные  Параметры:  n- Номер бита  Выходные параметры:  отсутствуют |
|  |  |  |  |
|  |  |  |  |
| ClrBit  Назначение:  Установить бит в ноль  Входные  Параметры:  n- Номер бита  Выходные параметры:  отсутствуют | GetBit  Назначение:  Получение значения бита  Входные  Параметры:  n- Номер бита  Выходные параметры:  Получение значения бита  (0 или 1) |  |  |

Операции

Вывод Ввод

Operator<< Operator>>

Назначение: Назначение:

Вывод битового поля ввод битового поля

Входные Входные

параметры: параметры:

ostream& ostr-ссылка на поток. Istream& istr-Ссылка на поток,

const TBitField& bf TBitFitField& bf-

Константная ссылка неконстантная ссылка

на битовое поле на битовое поле

Выходные Выходные

параметры: параметры:

поток с поток с введённой

битовым полем формата битовой строкой

(1010101 и т.д)

|  |
| --- |
|  |

Конструкторы/деструктор

|  |  |
| --- | --- |
| Конструктор  инициализатор  Назначение:  Создание битового поля  Входные параметры:  Len-Длина битового поля  Выходные параметры:  Отсутствуют | Конструктор  копирования  Назначение:  Копирование  битовых полей  Входные  Параметры:  Const TBitField& bf –  Константная ссылка на битовое поле  Выходные  параметры:  отсутствуют |

Деструктор

Назначение:

Освобождение памяти

Входные

параметры:

отсутствуют

Выходные

параметры:

### Описание класса TSet

class TSet

{

private:

int MaxPower;

TBitField BitField;

public:

TSet(int mp);

TSet(const TSet &s);

TSet(const TBitField &bf);

operator TBitField();

// доступ к битам

int GetMaxPower(void) const;

void InsElem(const int Elem);

void DelElem(const int Elem);

int IsMember(const int Elem) const;

// теоретико-множественные операции

int operator== (const TSet &s) const;

int operator!= (const TSet &s) const;

const TSet& operator=(const TSet &s);

TSet operator+ (const int Elem);

TSet operator- (const int Elem);

TSet operator+ (const TSet &s);

TSet operator\* (const TSet &s);

TSet operator~ (void);

TSet operator-(const TSet& obj);

friend istream &operator>>(istream &istr, TSet &bf);

friend ostream &operator<<(ostream &ostr, const TSet &bf);

};

Битовые поля:

MaxPower – максимальная мощность множества

TBitField – битовое поле

Методы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | |  | |  |
| GetMaxPower  Назначение:  Получение мощности множества  Входные параметры:  Отсутствуют  Выходные  Параметры:  Мощность множества | InsElem  Назначение:  Добавление элемента в множество  Входные параметры:  Elem- добавляемый элемент  Выходные  параметры:  отсутствуют | | DelElem  Назначение:  Исключение элемента из множества  Входные  Параметры:  Elem– удаляемый элемент  Выходные параметры:  отсутствуют | | IsMember  Назначение:  Проверка на принадлежность  Входные параметры:  Elem – элемент для проверки  Выходные параметры:  Значение бита  (0 или 1) |
|  |  | |  | |  |
|  |  | |  | |  |
|  | Операции | |  | |  |
| Равенство(==)  Operator==  Назначение:  Проверка на равенство двух множеств  Входные параметры:  s- множество  Выходные параметры:  Целое число  (0 или 1) | Неравенство(!=)  Operator!=  Назначение:  Проверка на неравенство двух множеств  Входные  параметры:  s – множество  Выходные  параметры:  Целое число  (0 или 1) | | Присваивание(=)  Назначение:  Присвоение значений полей одного объекта класса другому  Входные параметры:  s – множество  Выходные параметры:  Ссылка на объект  своего класса TSet | |  |
| Объединение с элементом  Operator+  Назначение:  Побитовое сложение элемента множества с элементом  Входные параметры:  Elem- добавляемый элемент  Выходные параметры:  Результирующее множество  Разность  с элементом  operator–  Назначение:  Исключение соответствующего элемента множества  Входные параметры:  Elem – вычитаемый элемент  Выходные параметры:  Результирующее множество | Пересечение  с элементом  operator&  Назначение:  Побитовое умножение соответствующего элемента множества с элементом  Входные  параметры:  Elem– добавляемый элемент  Выходные параметры:  Результирующее множество  Дополнение  к множеству  operator~  Назначение:  Инвертировать значения битового поля. Это и будет дополнение к множеству.  Входные  параметры:  отсутствуют  Выходные параметры:  Результирующее множество | | Пересечение множеств  Operator&  Назначение:  Побитовое умножение элементов двух множеств  Входные параметры:  s – множество  Выходные параметры:  Результирующее множество  Вывод  Operator<<  Назначение:  Вывод элементов множества в формате({e1,e2,…,en})  Входные параметры:  Ostream& ostr- ссылка на поток,  Const TSet& s-константная ссылка на объект класса TSet  Выходные параметры:  Поток с множеством формата(A={e1,e2,…,en} и т.д) | |  |
|  |  | |  | |  |
| Ввод  Operator>>  Назначение:  Заполнение множества элементами  Входные  параметры:  Istream& istr – ссылка на поток,TSet& s –ссылка на объект класса TSet |  | |  | |  |
| Конструктор инициализатор  Назначение:  Создание множеств  Входные параметры:  Mp – мощность множества  Выходные параметры:  Отсутствуют  Оператор преобразования  Operator TBitField()  Назначение:  Преобразование из TSet в TBitField  Входные параметры:  Отсутствуют  Выходные параметры:  Объект класса TBitField | Конструктор Копирования  Назначение:  Копирование множеств  Входные параметры:  s – множество  Выходные параметры:  Отсутствуют | | Конструктор преобразования типа:  Назначение:  Преобразование из TBitField в TSet  Входные параметры:  Bf – Битовое поле  Выходные параметры:  Отсутствуют | |  |
|  | |  | |

# Заключение

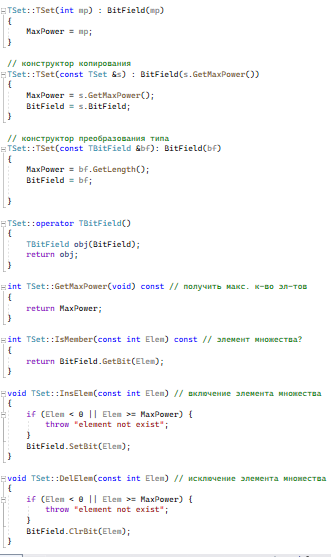
По результатам лабораторной работы были реализованы классы TSet и TBitField,а также написаны приложения и тесты для проверки работоспособности реализации. К лабораторной работе был составлен полный отчёт по теме со всеми подробными описаниями.

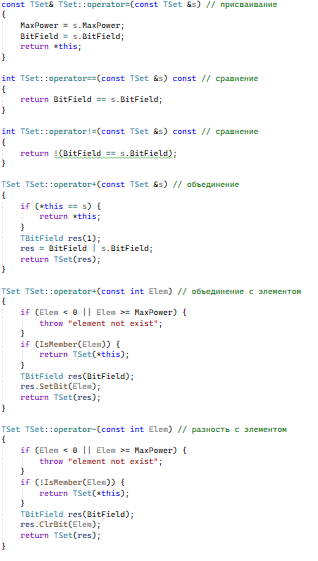
# Литература

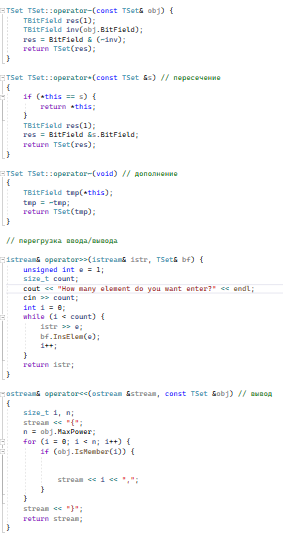
1. Битовые поля и операции над ними с.[33](https://rep.bntu.by/bitstream/handle/data/32621/Bitovye_polya_i_operacii_nad_nimi.pdf?sequence=1&isAllowed=y)
2. Битовые поля. [Урок 32](https://narodstream.ru/c-urok-32-bitovye-polya/)
3. Битовые поля [раздел Битовые поля](https://www.c-cpp.ru/books/bitovye-polya)

# Приложения

## Приложение А. Реализация класса TSet







## Приложение Б. Реализация класса TBitField

