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**

**ЛАБОРАТОРНАЯ РАБОТА**

**«Битовые поля»**

**Выполнил:**студент группы 381706-2

Гущин Александр Владимирович

\_\_\_\_\_\_\_\_\_\_\_  Подпись

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_  Подпись

Нижний Новгород

2018

Оглавление

[1.Введение 3](#_Toc532762702)

[2.Постановка задачи 4](#_Toc532762703)

[3.Руководство пользователя 5](#_Toc532762704)

[4.Руководство программиста 6](#_Toc532762705)

[4.1.Описание структуры программы 6](#_Toc532762706)

[4.2Описание алгоритмов 6](#_Toc532762707)

[5.Эксперименты 9](#_Toc532762708)

[6.Заключение 10](#_Toc532762709)

[7.Литература 11](#_Toc532762710)

[8.Приложения 12](#_Toc532762711)

[TBitField.ccp 12](#_Toc532762712)

[TSet.cpp 19](#_Toc532762713)

1.Введение

Битовые поля широко используется для хранения информации. Отличительной особенностью битовых полей является эффективное использование памяти. Благодаря этому этот тип хранения данных находит разнообразное применение в с сфере математических задач.

2.Постановка задачи

Необходимо написать реализацию битового поля и множества со всеми сопутствующими конструкторами, методами и перегрузками. Произвести замеры времени.

3.Руководство пользователя

При запуске программы пользователю предлагается ввести верхнюю границу.

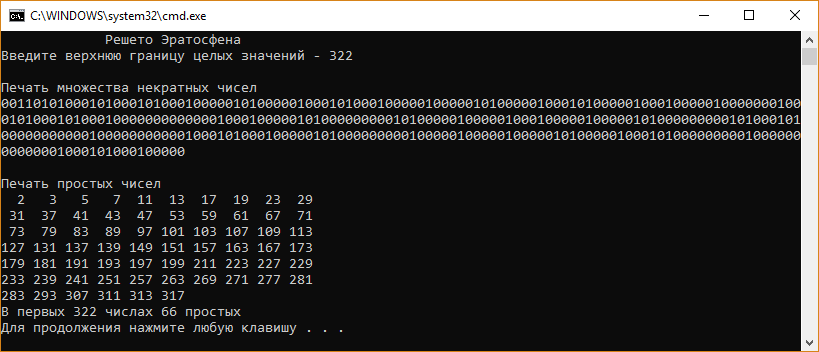


Рис. 1 запуск программы

4.Руководство программиста

4.1.Описание структуры программы

Программа состоит из таких модулей как

* tbitfield.cpp/ tbitfield.h – реализация битового поля
* tset.cpp/tset.h – реализация множества
* sample\_prime\_numbers.cpp – реализация решета Эратосфена

4.2.Описание структур данных

1. int GetMemIndex(const int n) const - метод класса. Возвращает нужный индекс в массиве из int.
2. TELEM GetMemMask (const int n) const - метод класса. Возвращает маску для бита n.
3. TBitField(int len) – конструктор с параметром. Принимает длину битового поля. Содержит проверку на отрицательность длины. Сначала выделяется память на нужное количество int. После чего производится обнуление.
4. TBitField(const TBitField &bf) – конструктор копирования. Сначала выделяется память на нужное количество int. После чего производится копирование. Сложность O(n).
5. ~TBitField() – деструктор. Удаляет массив из int.
6. int GetLength(void) const – метод класса. Возвращает длину битового поля.
7. void SetBit(const int n) - метод класса. Ставит 1 в битовое поле на позицию n. Содержит проверку на выход за пределы поля. Установка производится логическим | между нужным int и маской от n.
8. void ClrBit(const int n) - метод класса. Ставит 0 в битовое поле на позицию n. Содержит проверку на выход за пределы поля. Установка производится & между нужным int и отрицанием маски от n.
9. int GetBit(const int n) const - метод класса. Возвращает 0 или 1, смотря что стоит в битовом поле на позиции n. Содержит проверку на выход за пределы поля.
10. int operator==(const TBitField &bf) const – перегрузка операции == . Возвращает 0 если массивы различаются или 1 если они одинаковы. Сложность O(n).
11. int operator!=(const TBitField &bf) const - перегрузка операции != . Возвращает 1 если массивы различаются или 0 если они одинаковы. Сложность O(n).
12. TBitField& operator=(const TBitField &bf) - перегрузка операции =. Сначала перераспределяет память под новый размер. Затем приравнивает массивы. Сложность O(n).
13. TBitField operator|(const TBitField &bf) – перегрузка операции |. Производит операцию | между параллельными элементами из 2 массивов. После чего возвращает получившееся битовое поле. Содержит проверку длинны битовых полей. Сложность O(n).
14. TBitField operator&(const TBitField &bf) – перегрузка операции &. Производит операцию & между параллельными элементами из 2 массивов. После чего возвращает получившееся битовое поле. Содержит проверку длинны битовых полей. Сложность O(n).
15. TBitField operator~(void) перегрузка операции ~. Производит операцию ~ между элементами всеми элементами массива. После чего возвращает получившееся битовое поле. Сложность O(n).
16. friend istream &operator>>(istream &istr, TBitField &bf) – перегрузка ввода.
17. friend ostream &operator<<(ostream &ostr, const TBitField &bf) – перегрузка вывода.
18. TSet(int mp) - конструктор с параметром. Принимает длину битового поля. Содержит проверку на отрицательность длины. Выделяет память на нужное количество int.
19. TSet(const TSet &s) - конструктор копирования. Сначала выделяется память на нужное количество int. Затем производит копирование. Сложность O(n).
20. TSet(const TBitField &bf) - конструктор преобразования типа. Переводит TBitField в TSet. Сложность O(n).
21. operator TBitField() – преобразование типа. Переводит TBitField в TSet. Сложность O(n).
22. int GetMaxPower(void) const – возвращает мощность множества.
23. void InsElem(const int Elem) – метод. Добавляет элемент в множество.
24. void DelElem(const int Elem) – метод. Удаляет элемент из множества.
25. int IsMember(const int Elem) const – метод. Проверяет на наличие элемента в множестве.
26. int operator== (const TSet &s) const - перегрузка операции == . Возвращает 0 если множества различаются или 1 если они одинаковы. Сложность O(n).
27. int operator!= (const TSet &s) const - перегрузка операции != . Возвращает 1 если множества различаются или 0 если они одинаковы. Сложность O(n).
28. TSet& operator=(const TSet &s) - перегрузка операции =. Сначала перераспределяет память под новый размер. Затем приравнивает множества. Сложность O(n).
29. TSet operator+ (const int Elem) - объединение с элементом.
30. TSet operator- (const int Elem) - разность с элементом.
31. TSet operator+ (const TSet &s) – объединение множеств. Сложность O(n).
32. TSet operator\* (const TSet &s) – пересечение множеств.
33. TSet operator~ (void) – обратное множество. Сложность O(n).
34. friend istream &operator>>(istream &istr, TSet &bf) - перегрузка ввода.
35. friend ostream &operator<<(ostream &ostr, const TSet &bf) - перегрузка вывода.

Поля TBitField:

1. int bitLen - длина битового поля.
2. TELEM \*pMem - память для представления битового поля.
3. int memLen - к-во эл-тов Мем для представления бит.поля.

Поля TSet:

1. int maxPower - максимальная мощность множества.
2. TBitField bitField - битовое поле для хранения характеристического вектора.

4.3.Описание алгоритмов

TBitField operator~(void) – перегрузка операции отрицание. Сначала создается переменная T типа TBitField, после чего по циклу от нуля до memLen происходит присваивание i-тому элементу массива T, отрицание исходного i-того элемента массива. И далее происходит побитовый сдвиг налево и сразу направо элемента массива T с индексом memLen – 1 на кол-во элементов за пределами bitLen.

5.Эксперименты

Для эксперимента был взята логическая операция ИЛИ из TBitField.



рис. 2 график

Характеристики компьютера: Amd fx-8350, 8gb оперативной памяти

6.Заключение

Благодаря этой работе у меня появился опыт взаимодействия с битовыми полями. Битовые поля значительно уменьшают затраты на хранение информации и помогают решать многие практические задания.

7.Литература

1. Википедия <https://ru.wikipedia.org/wiki/Битовое_поле> (Дата обращения 15.12.2018)
2. Сайт c-cpp.ru <http://www.c-cpp.ru/books/bitovye-polya> (Дата обращения 15.12.2018)