Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Отчёт по лабораторной работе

Множества на основе битовых полей

Выполнил:

студент института ИТММ гр. 381908-4

Грушин Д.А.

Проверил:

ассистент каф. МОСТ, ИИТММ

Лебедев И.Г.

Нижний Новгород

2020 г.

Содержание

[Введение 3](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154854)

[Постановка задачи 4](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154855)

[Руководство пользователя 5](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154856)

[Руководство программиста 6](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154857)

[Описание структуры программы 6](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154858)

[Описание структур данных 6](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154859)

[Описание алгоритмов 8](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154860)

[Эксперименты 9](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154861)

[Заключение 11](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154862)

[Литература 12](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154863)

[Приложения 13](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154864)

[Приложение 1 13](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154865)

[Приложение 2 13](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154866)

[Приложение 3 13](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154867)

[Приложение 4 18](file:///C:\Users\грушины\Downloads\Отчёт%20по%20первой%20Лабораторной%20работе.doc#_Toc53154868)

# Введение

Данное задание предназначено для восстановления знаний, полученных с прошлых семестров и заполнения пробелов в них, если такие имеются.

В программе реализованы методы работы с множествами на основе битовых полей.

Было необходимо написать два класса согласно заданному интерфейсу, обеспечить работоспособность тестов, освоить GitHub и CMake, ознакомиться с Google Test.

Множество – это математический объект, являющийся совокупностью каких-либо объектов, которые называются элементами множества и обладают общим для всех их характеристическим свойством.

Некоторые операции над множествами:

* Пересечение,
* Объединение,
* Разность,
* Симметрическая разность.

Множества в программирование помогают решать задачи различного характера. Например, есть множество всех работников предприятия и множество работников, прошедших ежемесячную проверку. Тогда с легкостью возможно найти тех, кто из работников не прошел проверку.

# Постановка задачи

Цель данной 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.

Имея шаблон, который содержит интерфейсы классов битового поля и множества (h-файлы), готовый набор тестов для каждого из указанных классов, пример использования класса битового поля и множества для решения задачи поиска простых чисел с помощью алгоритма "Решето Эратосфена", выполнить работу, которая предполагает решение следующих задач:

* Реализация класса битового поля TBitField согласно заданному интерфейсу,
* Реализация класса множества TSet согласно заданному интерфейсу,
* Обеспечение работоспособности тестов и примера использования,
* Реализация нескольких простых тестов на базе Google Test,
* Публикация исходных кодов в личном репозитории на GitHub.

# Руководство пользователя

Пользователю нужно запустить файл sample\_prime\_numbers.exe.

Откроется консольное приложение для тестирования программ поддержки битового поля на основе решета Эратосфена.

Программа запросит пользователя ввести верхнюю границу целых значений. (Приложение 1)

После ввода числа, программа выполнит действия и выведет результат в консоль. (Приложение 2)

Для повторного выполнения потребуется перезапустить программу.

# Руководство программиста

## ***Описание структуры программы***

#include <iomanip> - в этом файле описаны манипуляторы при работе с потоковыми операциями.

#include "tbitfield.h" – подключение файла TBitField.h, который описывает работу с битовыми полями.

#include "tset.h" – подключение файла TSet.h, который описывает работу с множествами, на основе битовых полей.

В функции «main» происходит инициализация битового поля, затем его заполнение, проверка до sqrt(n) и удаление кратных, оставшиеся числа являются простыми (Решето Эратосфена). Далее происходит вывод информации в консоль о некратных и простых числах.

Код классов в приложении.

## ***Описание структур данных***

Программа состоит из нескольких основных классов – tbitfield, tset.

**tbitfield.cpp, заголовок tbitfield.h.**

Реализованы конструктор с параметром, конструктор копирования, деструктор.

Содержит поля:

* BitLen для хранения длины битового поля,
* TELEM \*pMem для предоставления памяти битовому полю,
* MemLen для количества элементов битового поля.

Реализует методы:

* GetMemIndex – получение индекса,
* GetMemMask – маска для бита,
* GetLength – получение длины,
* SetBit – установка бита,
* ClrBit – очистка бита,
* GetBit – получение бита.

Реализует операции:

* operator== - оператор сравнения,
* operator!= - оператор сравнения,
* operator= - оператор присваивания,
* operator| - операция «ИЛИ»,
* operator& - оператор «И»,
* operator~ - оператор отрицания,
* &operator>> -оператор ввода,
* &operator<< - оператор вывода.

**tset.cpp, заголовок tset.h**

Реализованы конструктор с параметром, конструктор копирования, конструктор преобразования типа.

Содержит поля:

* MaxPower для хранения максимальной мощности множества,
* TBitField BitField – битовое поле для хранения характеристического вектора.

Реализует методы:

* GetMaxPower – получение максимальной мощности множества,
* InsElem – включение элемента в множество,
* DelElem – удаление элемента из множества,

Реализует операции:

* operator== - оператор сравнения,
* operator!= - оператор сравнения,
* operator= - оператор присваивания,
* operator+ - операция объединение с элементом,
* operator- - операция разность с элементом,
* operator+ - операция объединение множеств,
* operator\* - операция пересечение множеств,
* operator~ - операция дополнения,
* &operator>> -оператор ввода,
* &operator<< - оператор вывода.

## ***Описание алгоритмов***

Создание множества:

* Инициализация битового поля размером, равным мощности множества
* Выделение памяти
* Заполнение элементов нулями

Добавление элемента в множество:

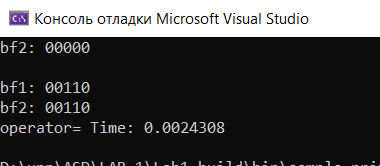
* Инициализация битового поля
* Передача элемента в класс битового поля,
* На основе элемента получаем индекс и маску,
* Используя побитовое «ИЛИ», присваиваем по полученному индексу, полученную маску.

Удаление элемента из множества:

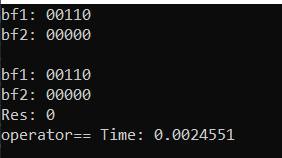
* Передача элемента в класс битового поля,
* На основе элемента получаем индекс и маску,
* Используя побитовое «И», присваиваем по полученному индексу, полученную маску, предварительно применив к маске побитовую инверсию.

# Эксперименты

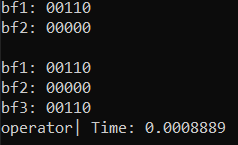
Результат выполнения операции присваивание:



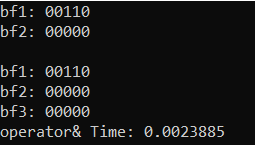
Результат выполнения операции сравнения:



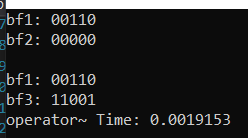
Результат операции “ИЛИ”:



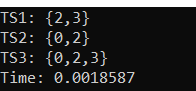
Результат операции “И”:



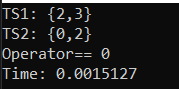
Результат операции отрицание:



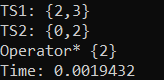
Результат объединения множеств:



Результат операции сравнения множеств:



Результат операции пересечения множеств:



Результат операции дополнение:



# Заключение

В ходе выполнения данной лабораторной работы были получены знания о Google Tets, работе с битовыми полями и множествами в языке программирования C++, с помощью которых была успешно реализована программа.

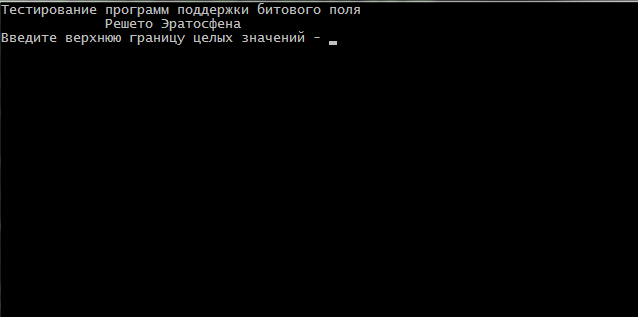
Так же был изучен ресурс GitHub и принципы работы с ним. Изучена утилита CMake.

# Литература

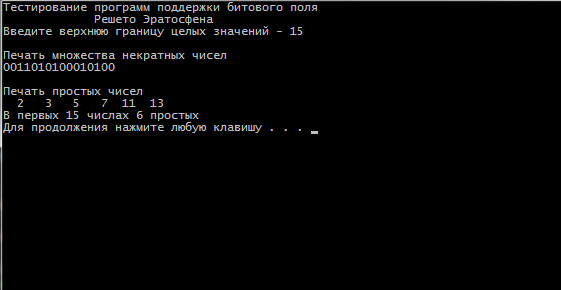
1. Столлингс, В. Структурная организация и архитектура компьютерных систем, 5-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2002. — 896 с.: ил. — Парал. тит. англ.
2. Johnson M. Superscalar Microprocessor Design. — Englewood Cliff, New Jersey: Prentice Hall, 1991.
3. Касперски К. Техника оптимизации 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.

# Приложения

Приложение 1:



Приложение 2:



Приложение 3:

Код TBitField.cpp:

// ННГУ, ВМК, Курс "Методы программирования-2", С++, ООП

//

// tbitfield.cpp - Copyright (c) Гергель В.П. 07.05.2001

// Переработано для Microsoft Visual Studio 2008 Сысоевым А.В. (19.04.2015)

//

// Битовое поле

#include "tbitfield.h"

TBitField::TBitField(int len)

{

if (len < 0)

{

throw "ERROR";

}

BitLen = len;

MemLen = len / (sizeof(TELEM) \* 8) + 1;

pMem = new TELEM[MemLen];

for (int i = 0; i < MemLen; i++)

{

pMem[i] = 0;

}

}

TBitField::TBitField(const TBitField &bf) // конструктор копирования

{

BitLen = bf.BitLen;

MemLen = bf.MemLen;

pMem = new TELEM[MemLen];

if (pMem != NULL)

{

for (int i = 0; i < MemLen; i++)

{

pMem[i] = bf.pMem[i];

}

}

}

TBitField::~TBitField()

{

if (pMem != NULL)

{

delete[]pMem;

pMem = NULL;

MemLen = NULL;

}

}

int TBitField::GetMemIndex(const int n) const // индекс Мем для бита n

{

return (n / (sizeof(TELEM) \* 8));

}

TELEM TBitField::GetMemMask(const int n) const // битовая маска для бита n

{

int index = (n % (sizeof(int) \* 8));

TELEM mask = 1 << index;

return mask;

}

// доступ к битам битового поля

int TBitField::GetLength(void) const // получить длину (к-во битов)

{

return BitLen;

}

void TBitField::SetBit(const int n) // установить бит

{

if (n < 0 || n > BitLen)

{

throw "ERROR";

}

pMem[GetMemIndex(n)] |= GetMemMask(n);

}

void TBitField::ClrBit(const int n) // очистить бит

{

if (n < 0 || n > BitLen)

{

throw "ERROR";

}

pMem[GetMemIndex(n)] &= ~GetMemMask(n);

}

int TBitField::GetBit(const int n) const // получить значение бита

{

if (n < 0 || n > BitLen)

{

throw "ERROR";

}

return pMem[GetMemIndex(n)] & GetMemMask(n);

}

// битовые операции

TBitField& TBitField::operator=(const TBitField &bf) // присваивание

{

if (this == &bf)

{

return \*this;

}

BitLen = bf.BitLen;

MemLen = bf.MemLen;

if (bf.MemLen != MemLen)

{

MemLen = bf.MemLen;

delete pMem;

pMem = new TELEM[MemLen];

}

for (int i = 0; i < MemLen; i++)

{

pMem[i] = bf.pMem[i];

}

return \*this;

}

int TBitField::operator==(const TBitField &bf) const // сравнение

{

if (BitLen != bf.BitLen)

{

return 0;

}

else

{

for (int i = 0; i < MemLen; i++)

{

if (pMem[i] != bf.pMem[i])

{

return 0;

break;

}

}

}

return 1;

}

int TBitField::operator!=(const TBitField &bf) const // сравнение

{

if (BitLen == bf.BitLen)

{

return 1;

}

else

{

for (int i = 0; i < MemLen; i++)

{

if (pMem[i] == bf.pMem[i])

{

return 1;

break;

}

}

}

return 0;

}

TBitField TBitField::operator|(const TBitField &bf) // операция "или"

{

int len = BitLen;

if (bf.BitLen > len)

{

len = bf.BitLen;

}

TBitField temp(len);

for (int i = 0; i < MemLen; i++)

{

temp.pMem[i] = pMem[i];

}

for (int i = 0; i < bf.MemLen; i++)

{

temp.pMem[i] |= bf.pMem[i];

}

return temp;

}

TBitField TBitField::operator&(const TBitField &bf) // операция "и"

{

int i, len = BitLen;

if (bf.BitLen > len)

{

len = bf.BitLen;

}

TBitField temp(len);

for (i = 0; i < MemLen; i++)

{

temp.pMem[i] = pMem[i];

}

for (i = 0; i < bf.MemLen; i++)

{

temp.pMem[i] &= bf.pMem[i];

}

if (bf.BitLen < len)

{

for (int i = bf.BitLen; i < MemLen; i++)

{

temp.pMem[i] = 0;

}

}

return temp;

}

TBitField TBitField::operator~(void) // отрицание

{

TBitField tmp(\*this);

for (int i = 0; i < tmp.BitLen; i++)

{

if (tmp.GetBit(i))

{

tmp.ClrBit(i);

}

else

{

tmp.SetBit(i);

}

}

return tmp;

}

// ввод/вывод

istream &operator>>(istream &istr, TBitField &bf) // ввод

{

int i = 0; char ch;

do

{

istr >> ch;

} while (ch != ' ');

while (1)

{

istr >> ch;

if (ch == '0')

{

bf.ClrBit(i++);

}

else if (ch == '1')

{

bf.SetBit(i++);

}

else break;

}

return istr;

}

ostream &operator<<(ostream &ostr, const TBitField &bf) // вывод

{

int len = bf.GetLength();

for (int i = 0; i < len; i++)

{

if (bf.GetBit(i))

{

ostr << '1';

}

else

{

ostr << '0';

}

}

return ostr;

}

Код tset.cpp:

#include "tset.h"

TSet::TSet(int mp) : BitField(mp)

{

MaxPower = mp;

}

// конструктор копирования

TSet::TSet(const TSet &s) : BitField(s.BitField)

{

{

MaxPower = s.MaxPower;

}

}

// конструктор преобразования типа

TSet::TSet(const TBitField& bf) :MaxPower(bf.GetLength()), BitField(bf)

{

}

TSet::operator TBitField()

{

return BitField;

}

int TSet::GetMaxPower(void) const // получить макс. к-во эл-тов

{

return MaxPower;

}

int TSet::IsMember(const int Elem) const // элемент множества?

{

return BitField.GetBit(Elem);

}

void TSet::InsElem(const int Elem) // включение элемента множества

{

BitField.SetBit(Elem);

}

void TSet::DelElem(const int Elem) // исключение элемента множества

{

BitField.ClrBit(Elem);

}

// теоретико-множественные операции

TSet& TSet::operator=(const TSet &s) // присваивание

{

if (this == &s) //Проверка на самоприсваивание

{

return \*this;

}

BitField = s.BitField;

MaxPower = s.GetMaxPower();

return \*this;

}

int TSet::operator==(const TSet &s) const // сравнение

{

return BitField == s.BitField;

}

int TSet::operator!=(const TSet &s) const // сравнение

{

return BitField != s.BitField;

}

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

{

TSet temp(BitField | s.BitField);

return temp;

}

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

{

BitField.SetBit(Elem);

return \*this;

}

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

{

BitField.ClrBit(Elem);

return \*this;

}

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

{

TSet temp(BitField & s.BitField);

return temp;

}

TSet TSet::operator~(void) // дополнение

{

TSet temp(~BitField);

return temp;

}

// перегрузка ввода/вывода

istream &operator>>(istream &istr, TSet &s) // ввод

{

int temp; char ch;

do

{

istr >> ch;

} while (ch != '{');

do {

istr >> temp;

s.InsElem(temp);

do

{

istr >> ch;

} while ((ch != ',') && (ch != '}'));

} while (ch != '}');

return istr;

}

ostream& operator<<(ostream &ostr, const TSet &s) // вывод

{

int i, n; char ch = '{';

n = s.GetMaxPower();

for (i = 0; i < n; i++)

{

if (s.IsMember(i))

{

ostr << ch << i;

ch = ',';

}

}

ostr << "}";

return ostr;

}

Код функции main:

// ННГУ, ВМК, Курс "Методы программирования-2", С++, ООП

//

// sample\_prime\_numbers.cpp - Copyright (c) Гергель В.П. 20.08.2000

// Переработано для Microsoft Visual Studio 2008 Сысоевым А.В. (19.04.2015)

//

// Тестирование битового поля и множества

#include <iomanip>

#ifndef USE\_SET // Использовать класс TBitField

#include "tbitfield.h"

int main()

{

int n, m, k, count;

setlocale(LC\_ALL, "Russian");

cout << "Тестирование программ поддержки битового поля" << endl;

cout << " Решето Эратосфена" << endl;

cout << "Введите верхнюю границу целых значений - ";

cin >> n;

TBitField s(n + 1);

// заполнение множества

for (m = 2; m <= n; m++)

s.SetBit(m);

// проверка до sqrt(n) и удаление кратных

for (m = 2; m \* m <= n; m++)

// если m в s, удаление кратных

if (s.GetBit(m))

for (k = 2 \* m; k <= n; k += m)

if (s.GetBit(k))

s.ClrBit(k);

// оставшиеся в s элементы - простые числа

cout << endl << "Печать множества некратных чисел" << endl << s << endl;

cout << endl << "Печать простых чисел" << endl;

count = 0;

k = 1;

for (m = 2; m <= n; m++)

if (s.GetBit(m))

{

count++;

cout << setw(3) << m << " ";

if (k++ % 10 == 0)

cout << endl;

}

cout << endl;

cout << "В первых " << n << " числах " << count << " простых" << endl;

}

#else

#include "tset.h"

int main()

{

int n, m, k, count;

setlocale(LC\_ALL, "Russian");

cout << "Тестирование программ поддержки множества" << endl;

cout << " Решето Эратосфена" << endl;

cout << "Введите верхнюю границу целых значений - ";

cin >> n;

TSet s(n + 1);

// заполнение множества

for (m = 2; m <= n; m++)

s.InsElem(m);

// проверка до sqrt(n) и удаление кратных

for (m = 2; m \* m <= n; m++)

// если м в s, удаление кратных

if (s.IsMember(m))

for (k = 2 \* m; k <= n; k += m)

if (s.IsMember(k))

s.DelElem(k);

// оставшиеся в s элементы - простые числа

cout << endl << "Печать множества некратных чисел" << endl << s << endl;

cout << endl << "Печать простых чисел" << endl;

count = 0;

k = 1;

for (m = 2; m <= n; m++)

if (s.IsMember(m))

{

count++;

cout << setw(3) << m << " ";

if (k++ % 10 == 0)

cout << endl;

}

cout << endl;

cout << "В первых " << n << " числах " << count << " простых" << endl;

}

#endif