*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(национальный исследовательский университет)***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_Компьютерные Системы и сети (ИУ6)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № 7**

**Название лабораторной работы: Программирование с использованием контейнеров STL и Qt**

**Дисциплина: Объектно-ориентированное программирование**

Студент гр. ИУ6-22Б  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Плютто**



(Подпись, дата) (И.О. Фамилия)



Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Веселовская**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Задание

Организовать программным способом символьные файлы ***F*** и ***G.*** С помощью множеств stl::set и QSet определить, совпадают ли слова в этих файлах. Выдать список слов, которые не совпадают. Реализовать свои функции сравнения, игнорирующие различия в регистре символов. Сравнить скорость работы разных реализаций в отладочном и финальном режиме.

Решение

*#include* <QCoreApplication>

*#include* <QSet>

*#include* <QString>

*#include* <QFile>

*#include* <set>

*#include* <string>

*#include* <cstdlib>

*#include* <ctime>

*#include* <iostream>

*#include* <chrono>

*#include* <QElapsedTimer>

using std::set, std::string, std::rand;

QString generateWords(){

srand(time(NULL));

*char* alphabet[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

QString word = "";

*for* (*int* j = 0; j < arc4random() % 10000; j++){

*int* length = arc4random() % 10 + 1;

*for* (*int* i = 0; i < length; i++) {

*int* index = arc4random() % 26;

*char* letter = alphabet[index];

word += letter;

}

word += ' ';

}

*return* word;

}

*void* dofiles(){

QFile fileF("F");

QFile fileG("G");

*if* (fileF.open(QIODevice::WriteOnly | QIODevice::Text) &&

fileG.open(QIODevice::WriteOnly | QIODevice::Text)){

QTextStream writeStreamF(&fileF);

QTextStream writeStreamG(&fileG);

writeStreamF << generateWords();

writeStreamG << generateWords();

fileF.close();

fileG.close();

}

}

*int* main(*int* argc, *char* \*argv[])

{

QCoreApplication a(argc, argv);

set<string> fsset;

set<string> gsset;

QSet<QString> fqset;

QSet<QString> gqset;

dofiles();

QFile fileF("F");

QFile fileG("G");

*if* (fileF.open(QIODevice::ReadOnly | QIODevice::Text) &&

fileG.open(QIODevice::ReadOnly | QIODevice::Text)){

QTextStream readStreamF(&fileF);

QTextStream readStreamG(&fileG);

QString Qfstr = readStreamF.readLine();

QString Qgstr = readStreamG.readLine();

string fstr = Qfstr.toStdString();

string gstr = Qgstr.toStdString();

QElapsedTimer timer1;

timer1.start();

*int* last{};

*for* (*int* i = 0; i < fstr.length(); i++){

*if* (fstr[i] == ' '){

fsset.insert(fstr.substr(last+1, i - last - 1));

last = i;

};

}

last = 0;

*for* (*int* i = 0; i < gstr.length(); i++){

*if* (fstr[i] == ' '){

gsset.insert(gstr.substr(last+1, i - last - 1));

last = i;

};

}

*for* (const *auto*& element : fsset) {

*if* (gsset.find(element) == gsset.end()) {

qDebug() << QString::fromStdString(element) << " ";

}

}

*int* t1 = timer1.nsecsElapsed();

QElapsedTimer timer2;

timer2.start();

last = 0;

*for* (*int* i = 0; i < Qfstr.length(); i++){

*if* (fstr[i] == ' '){

fqset.insert(Qfstr.mid(last+1, i - last - 1));

last = i;

};

}

last = 0;

*for* (*int* i = 0; i < Qgstr.length(); i++){

*if* (fstr[i] == ' '){

gqset.insert(Qgstr.mid(last+1, i - last - 1));

last = i;

};

};

qDebug() << '\n';

*for* (const *auto*& element : fqset) {

*if* (gqset.find(element) == gqset.end()) {

qDebug() << element;

}

}

fileF.close();

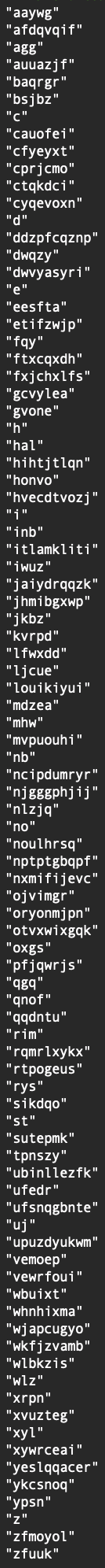
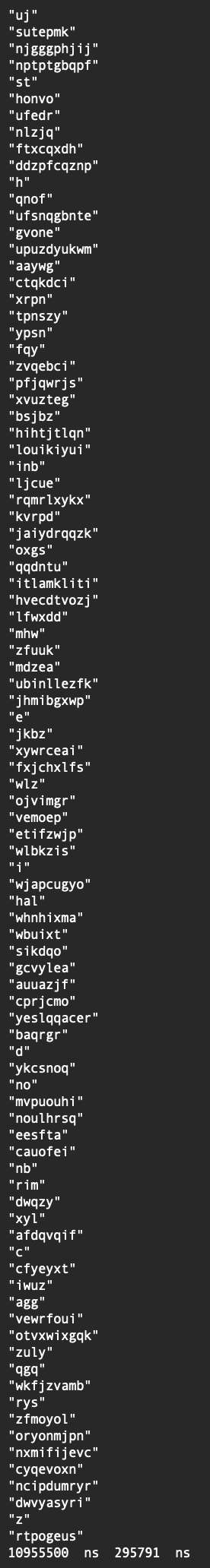
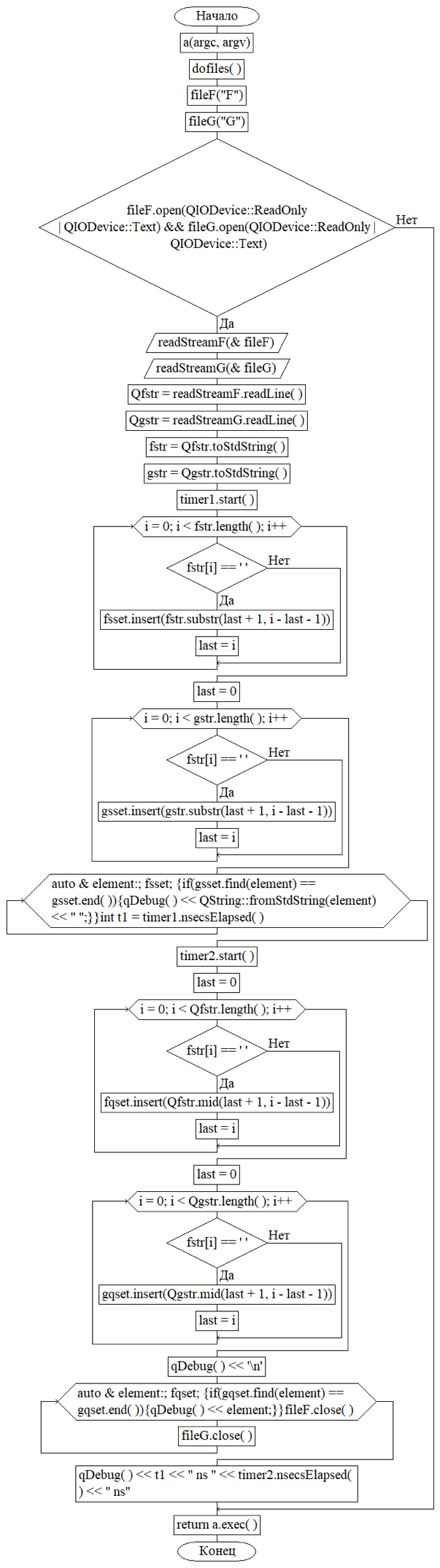
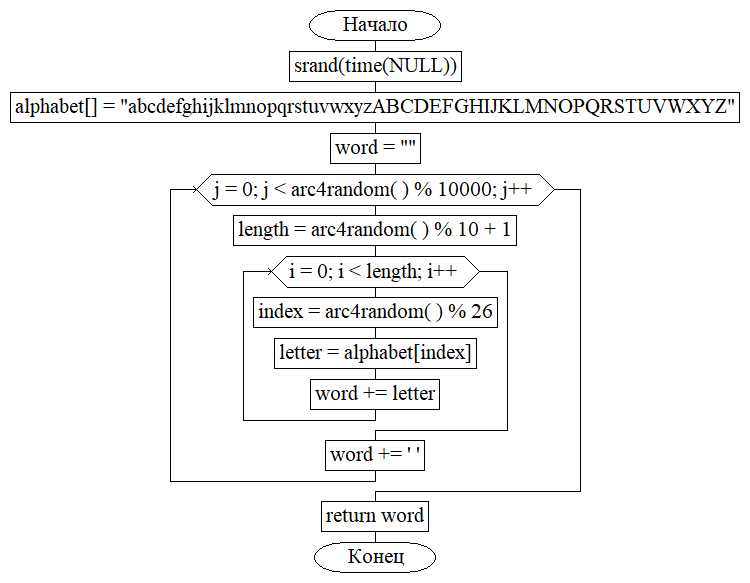
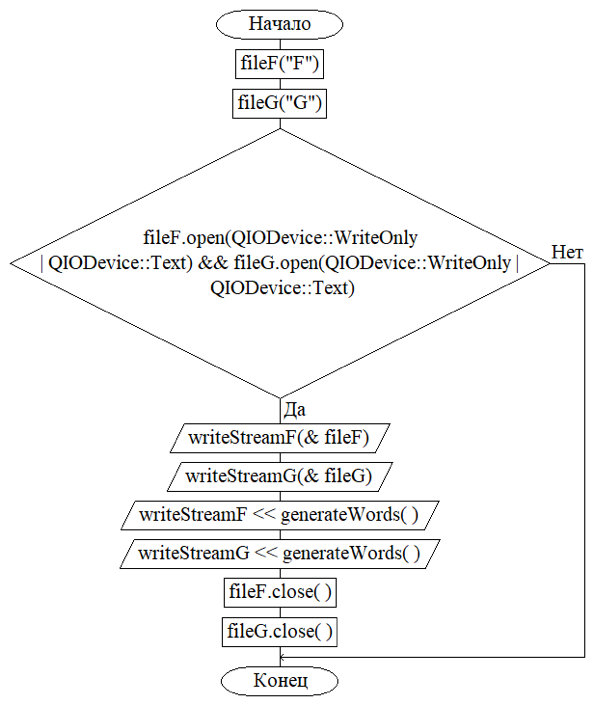
fileG.close();

qDebug() << t1 << " ns " << timer2.nsecsElapsed() << " ns";

}

*return* a.exec();

}

Вывод: я научился программировать с использованием контейнеров STL и QT.