**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Криптография»

**Лабораторная работа № 1**

Тема: факторизация чисел

Студент: Хренов Геннадий

Группа: 80-307Б

Преподаватель: Борисов А. В.

Дата:

Оценка:

Москва, 2021

1. Постановка задачи

Разложить каждое из чисел n1 и n2 на нетривиальные сомножители.

n1=119760639583941053725652803731328419697649739176243841021915621242807618608591,

n2=1916242087180680156861712994509728052535159091128844805658679025296716559404434664811725619186652725901325774649017594144788360637407178476936316915220758144535681964371311657071750970414707218112222280453951875213591639735019844579642622014874212594838041457800464921182345127496460888250084171815540351211745813542192969624108567504481905290317359415752535077985931507909722167364312980099834023023021212767107040301344392783417575981002593796696074442689507301

1. Метод решения

Вначале я попробовал вероятностные методы типа Полларда p-1 и Полларда p-0, которые не дали нужных результатов. Затем я приступил к реализации квадратичного решета на языке С++, используя довольно быструю библиотеку длинной арифметики gmp, в написании которой используется ассемблер. Программа показала хорошие результаты по факторизации 100-битовых чисел (в районе секунды или меньше в зависимости от числа), однако для заданных чисел такая реализация не подходит по времени. Разложить первое число удалось с помощью msieve – библиотека на Си для факторизации больших чисел, которая содержит реализацию алгоритмов SIQS и GNFS. Второе число длиной в 463 цифры обычным алгоритмом разложить за небольшое время невозможно. Но один из множителей этого числа можно найти как НОД с числом из другого варианта. В моем случае это число варианта 12. Второй множитель находим делением начального числа на первый.

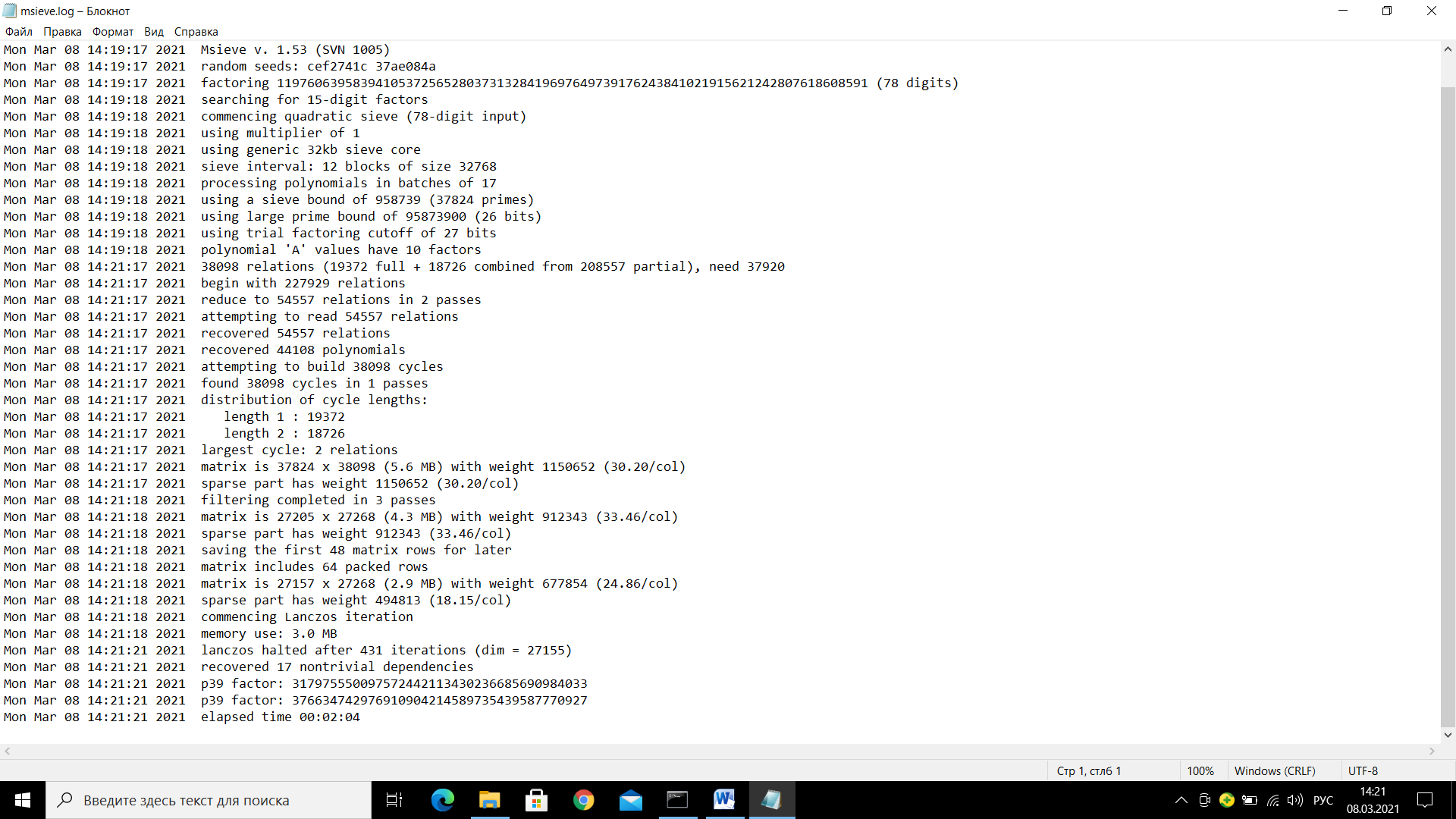
1. Структура программы

lab1.cpp - пробная реализация квадратичного решета

secondFactor.cpp – разложение второго числа

1. Результаты работы

Разложение первого числа с помощью msieve



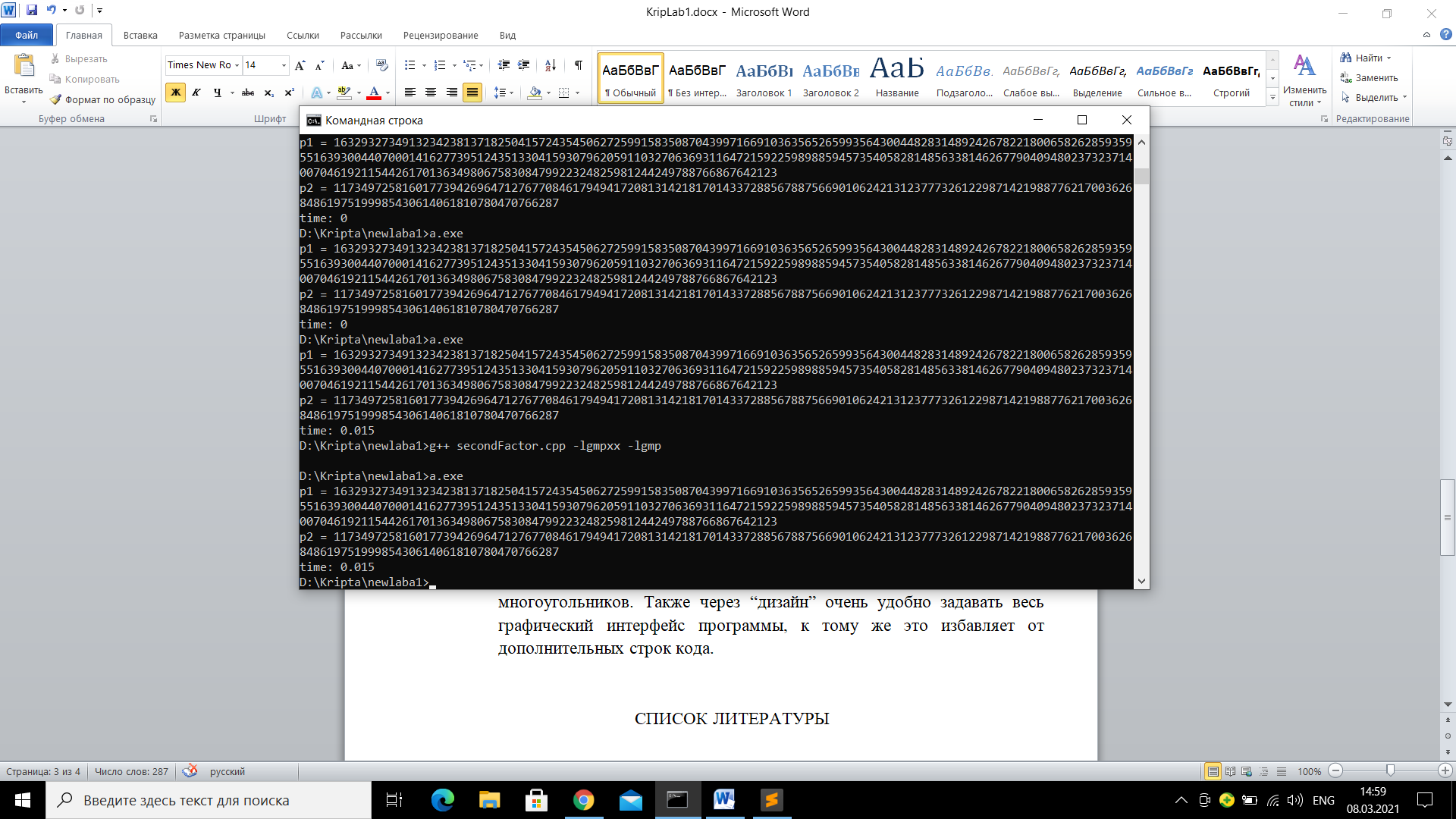
В итоге получено разложение:

P1: 317975550097572442113430236685690984033

P2: 376634742976910904214589735439587770927

За время: 2 минуты 4 секунды

Разложение второго числа



В итоге получено разложение:

P1: 163293273491323423813718250415724354506272599158350870439971669103635652659935643004482831489242678221800658262859359551639300440700014162773951243513304159307962059110327063693116472159225989885945735405828148563381462677904094802373237140070461921154426170136349806758308479922324825981244249788766867642123

P2: 11734972581601773942696471276770846179494172081314218170143372885678875669010624213123777326122987142198877621700362684861975199985430614061810780470766287

За время: 0.015 секунд

Характеристики ЭВМ

Процессор: AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx 2.30 GHz

Оперативная память: 8.00ГБ

Тип системы: 64-разрядная операционная система, процессор x64

1. Выводы

В ходе работы я познакомился с алгоритмами факторизации чисел. Оказалось, что для больших чисел это довольно сложная задача и для ее решения потребовалось либо использовать специальные методы со всеми возможными оптимизациями времени работы стандартных алгоритмов, либо вовсе не использовать стандартные алгоритмы и искать обходные пути. Такую задачу по факторизации с достаточно длинным числом нельзя решить каким-то стандартным способом за малый промежуток времени, поэтому простые числа используются в криптографии.

СПИСОК ЛИТЕРАТУРЫ

1. [Алгоритмы](https://www.bogotobogo.com/Qt/Qt5_OpenGL_QGLWidget.php) факторизации

<https://e-maxx.ru/algo/factorization>

2. Квадратичное решето

<https://habr.com/ru/post/521876/>