# Министерство образования и науки Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра теоретической и прикладной информатики

### Лабораторная работа № 7 по дисциплине «Основы теории информации и криптографии»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Факультет: | ПМИ |  |  |
| Группа: | ПМИ-62 |  |  |
| Студенты: | Трухин И.А. |  |  |
| Вариант: | 1 |  |  |
| Преподаватель: | Авдеенко Т. В. |  |  |

Новосибирск

2018

**Задание**

1. Реализовать приложение, удовлетворяющее следующим требованиям:
   1. Во входном файле хранятся входные данные, необходимые для работы программы (например, подлежащее факторизации число).
   2. Программа проверяет заданное число на простоту с помощью теста на простоту, реализованного в лабораторной работе № 6. Если оно является простым, то процедура факторизации не выполняется.
   3. Программа находит разложение заданного числа на произведение простых множителей.
   4. Программа выдаёт список простых делителей заданного числа с указанием степени, с которой они входят в разложение числа, время и количество итераций основного цикла, потребовавшихся для разложения.
2. С помощью реализованного приложения выполнить следующие задания:
   1. Протестировать правильность работы приложения на числах разной длины.
   2. Сделать выводы о проделанной работе.

|  |  |
| --- | --- |
| № варианта | Метода факторизации |
| 1 | Метод Ферма |

**Описание метода решения заданий**

Пусть n – подлежащее факторизации число, multipliers – коллекция «ключ - значение» множителей, a – один из простых множителей числа n, Ferma(x) – функция, возвращающая простой множитель целого числа x. Тогда алгоритм факторизации числа n методом Ферма следующий:

Вход: n

Выход: multipliers

multipliers 🡨

ПОКА n – составное число:

a 🡨 Ferma(n)

ЕСЛИ a нет в multipliers, ТО:

добавить пару (a, 1) в multipliers

ИНАЧЕ:

multipliers[a] = multipliers[a] +1

КОНЕЦ ЕСЛИ

n = n / a

КОНЕЦ ПОКА

ЕСЛИ n нет в multipliers, ТО:

добавить пару (n, 1) в multipliers

ИНАЧЕ:

multipliers[n] = multipliers[n] +1

**Описание проведенных исследований**

В таблице 1 приведены результаты проведенных исследований зависимости времени, требуемого для факторизации числа, от его разрядности, а также число итераций основного цикла.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| Разрядность числа | Число, подлежащее факторизации | Число итераций основного цикла | Время выполнения (в миллисекундах) |
| 3 | 345 | 2 | 4 |
| 5 | 34578 | 4 | 3 |
| 7 | 3457867 | 2 | 3 |
| 9 | 345786713 | 1 | 7 |

**Программный код на языке C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Numerics;

using System.Security.Cryptography;

using System.Diagnostics;

using System.IO;

namespace RabinMillerTest

{

class Program

{

static void Main(string[] args)

{

try

{

// счетчик числа итераций

int count\_iter = 0;

// чтение числа из файла

StreamReader sr = new StreamReader("C:\\Users\\User\\source\\repos\\RabinMillerTest\\RabinMillerTest\\input.txt");

string nstr = sr.ReadLine();

if (nstr == null) throw new Exception("Файл пуст");

// преобразуем считанную строку в число

int n = Convert.ToInt32(nstr);

if (n < 0) throw new Exception("Слишком малое значение n! (n > 0)");

//int n = 82944;

Dictionary<int, int> multipliers = new Dictionary<int, int>();

// выполняем замер времени

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

while (!Test(n, 10) && n > 1)

{

count\_iter++;

int a = Ferma(n);

if (multipliers.ContainsKey(a) == false) { multipliers.Add(a, 1); }

else { multipliers[a]++; }

n /= a;

}

if (multipliers.ContainsKey(n) == false) { multipliers.Add(n, 1); }

else { multipliers[n]++; }

stopWatch.Stop();

TimeSpan ts = stopWatch.Elapsed;

// вывод множителей и их степеней на консоль

foreach (KeyValuePair<int, int> keyValue in multipliers)

{

Console.WriteLine("Множитель - " + keyValue.Key + "; степень - " + keyValue.Value);

}

Console.WriteLine("Для факторизации числа потребовалось {0} итераций", count\_iter);

Console.WriteLine("Затраченное время составляет {0} миллисекунд", ts.Milliseconds);

Console.ReadKey();

}

catch (Exception E)

{

Console.WriteLine("\n\*\*\* Error! \*\*\*");

Console.WriteLine("Message: {0}\n", E.Message);

}

}

// производится k раундов проверки числа n на простоту

static bool Test(BigInteger n, int k){ . . . }

static int Ferma(int n)

{

if (n % 2 == 0) return 2;

if (n % 3 == 0) return 3;

if (n % 5 == 0) return 5;

if (n % 7 == 0) return 7;

if (n % 11 == 0) return 11;

if (n % 13 == 0) return 13;

if (n % 17 == 0) return 17;

if (n % 19 == 0) return 19;

for (int x = (int) Math.Sqrt(n); x <= n/2 + 1; x++){

// t = x^2 - n

int t = x\*x - n;

if (t >= 0) {

// if t = y^2

int y = (int)Math.Sqrt(t);

if (t == y\*y) { return x - y; }

}

}

return 1;

}

}

}

**Тестирование программ**

На рисунке 1 приведены результаты работы программы для числа 345786713.

