# 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 № 6 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, разрядность простого числа, вероятность признать составное число простым).
   2. Программа генерирует простое число с помощью заданного в варианте теста простоты.
   3. Программа выдаёт сгенерированное число, время и количество итераций основного цикла, потребовавшихся для генерации.
2. С помощью реализованного приложения выполнить следующие задания:
   1. Протестировать правильность работы приложения на числах разной длины.
   2. Сделать выводы о проделанной работе.

|  |  |
| --- | --- |
| № варианта | Тест |
| 1 | Тест Рабина-Миллера |

**Описание метода решения заданий**

Вероятностный тест простоты:

1. Выбираем случайное число *a* из интервала [1; *n*–1] и проверяем с помощью алгоритма Евклида условие НОД (*a*, *n*) = 1.
2. Если оно не выполняется, то ответ “*n* – составное”.
3. Вычисляем *z* = *at* mod *n*.
4. Если z ≡ ±1 (mod *n*), то переходим к шагу 1.
5. Вычисляем mod *n*,  mod *n*, … ,  mod *n* до тех пор, пока не появится –1.
6. Если ни одно из этих чисел не равно –1, то ответ “*n* – составное”;
7. Если мы достигли *n*, то ответ неизвестен (и тест можно повторить еще раз).

**Описание проведенных исследований**

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

Таблица 1

|  |  |  |
| --- | --- | --- |
| Разрядность числа | Число итераций основного цикла | Время выполнения (в миллисекундах) |
| 3 | 4 | 4 |
| 5 | 4 | 4 |
| 7 | 20 | 4 |
| 9 | 9 | 4 |
| 10 | 6 | 4 |

Как видно из таблицы 1, тест Рабина-Миллера позволяет быстро проверить число на простоту, что в свою очередь позволяет быстро генерировать большие простые числа, независимо от разрядности числа.

**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 разрядности 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 < 3) throw new Exception("Слишком малое значение n! (n > 2)");

// вычисляем верхнюю и нижнюю границу числа

BigInteger lowerline = (BigInteger)Math.Pow(10, n - 1) + 1;

BigInteger upperline = lowerline \* 2 - 11;

// выбиираем случаное число из возможного диапозона

Random rnd = new Random();

lowerline = rnd.Next((int)lowerline, (int)upperline);

// выполняем замер времени

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

for (BigInteger i = lowerline; i <= upperline; i++)

{

count\_iter++;

if(Test(i, 10))

{

Console.WriteLine(i);

break;

}

}

stopWatch.Stop();

TimeSpan ts = stopWatch.Elapsed;

Console.WriteLine("Для генерации простого 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)

{

// если n == 2 или n == 3 - эти числа простые, возвращаем true

if (n == 2 || n == 3)

return true;

// если n < 2 или n четное - возвращаем false

if (n < 2 || n % 2 == 0)

return false;

// представим n − 1 в виде (2^s)·t, где t нечётно, это можно сделать последовательным делением n - 1 на 2

BigInteger t = n - 1;

int s = 0;

while (t % 2 == 0)

{

t /= 2;

s += 1;

}

// повторить k раз

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

{

// выберем случайное целое число a в отрезке [2, n − 2]

RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();

byte[] \_a = new byte[n.ToByteArray().LongLength];

BigInteger a;

do

{

rng.GetBytes(\_a);

a = new BigInteger(\_a);

}

while (a < 2 || a >= n - 2);

// x ← a^t mod n, вычислим с помощью возведения в степень по модулю

BigInteger x = BigInteger.ModPow(a, t, n);

// если x == 1 или x == n − 1, то перейти на следующую итерацию цикла

if (x == 1 || x == n - 1) { continue; }

// повторить s − 1 раз

for (int r = 1; r < s; r++)

{

// x = x^2 mod n

x = BigInteger.ModPow(x, 2, n);

// если x == 1, то вернуть "составное"

if (x == 1) { return false; }

// если x == n − 1, то перейти на следующую итерацию внешнего цикла

if (x == n - 1) { break; }

}

if (x != n - 1) { return false; }

}

// вернуть "вероятно простое"

return true;

}

}

}

**Тестирование программ**

В таблице 2 приведены входные данные (разрядность числа), и вывод программы.

Таблица 2

|  |  |  |
| --- | --- | --- |
| Разрядность числа | Назначение | Вывод программы |
| -1 | Недопустимое значение разрядности |  |
| 7 | Тривиальный тест |  |
| 13 | Значение разрядности превышает допустимый интервал целочисленного типа данных (10 знаков) |  |