Федеральное государственное автономное образовательное учреждение высшего образования «**Уральский федеральный университет** имени первого Президента России Б. Н. Ельцина»

Отчет по лабораторной работе №1

дисциплины «Алгоритмы и анализ сложности»

Студент: Волков Илья Евгеньевич  
Группа: РИ-200004 (АТ-08)  
Преподаватель: Рыбкина Мария Николаевна

2022 год.

Оглавление

[Задание 3](#_Toc98096170)

[Теоретическая часть 3](#_Toc98096171)

[Инструкция пользователя 4](#_Toc98096172)

[Инструкция программиста 4](#_Toc98096173)

[Тестирование 5](#_Toc98096174)

[Вывод 7](#_Toc98096175)

# Задание

Лабораторная работа №1

Целочисленная арифметика произвольной точности и RSA-шифрование.  
  
1. Написать класс, который содержит целое число со знаком в виде массива однобайтовых элементов. Реализовать конструкторы, деструктор, перегрузить операции: аддитивные (+, -), мультипликативные (\*, /, %), сравнения (==, !=, <, >).  
  
2. Написать функцию шифрования строки с помощью простого числа (алгоритм типа RSA).  
  
3. Зашифровать/расшифровать текстовый файл с помощью открытого RSA-ключа.

# Теоретическая часть

Есть несколько способов решения данной задачи:

1. Найти готовую библиотеку. Я не выбрал этот способ, потому что скорее всего так выполнять задание нельзя.

2. Найти готовое решение и, основываясь на нем, написать свое. Я использовал этот способ, я нашел хорошее описание RSA и сделал по этому объяснению (в примере не было готового кода, только описание RSA и рекомендации к написанию своей реализации). Битовое представление целого числа я написал полностью самостоятельно.

3. Полностью написать все с нуля самостоятельно. Я не выбрал этот вариант, ибо я не знаком с принципом RSA шифрования.

# Инструкция пользователя

Для запуска софта используется файл Lab1.exe.   
Для введения текста для шифрования, используется input.txt.  
В файле keysDnN.txt хранятся закрытые ключи шифрования.  
В файле result.txt хранится расшифрованная строка.  
В файле output.txt хранится зашифрованная строка.  
  
Для шифрования:  
  
1. Ввести данные в input.txt

2. Запустить Lab1.exe

3. Готово. Результат в output.txt

Для дешифрования:

1. Ввести данные в output.txt

2. Запустить Lab1.exe

3. Готово. Результат в result.txt

# Инструкция программиста

Для запуска софта используется файл Lab1.exe.   
Для введения текста для шифрования, используется input.txt.  
В файле keysDnN.txt хранятся закрытые ключи шифрования.  
В файле result.txt хранится расшифрованная строка.  
В файле output.txt хранится зашифрованная строка.

ByteInt.cs – реализация байтового представления целого числа.

RSA.cs – реализация RSA шифрования.

Program.cs – мастер файл.

# Тестирование

Тестирование битового представления целого числа произведено в Program.cs:  
  
namespace Lab1;  
  
public class Main1  
{  
 private static void Main()  
 {  
 *// Пример работы ByteInt:* var bi = new ByteInt(456);  
 var bi1 = new ByteInt(874);  
   
 Console.WriteLine($"{bi.Convert2Int()} + {bi1.Convert2Int()} = {(bi + bi1).Convert2Int()}");  
 Console.WriteLine($"{bi.Convert2Int()} - {bi1.Convert2Int()} = {(bi - bi1).Convert2Int()}");  
 Console.WriteLine($"{bi.Convert2Int()} \* {bi1.Convert2Int()} = {(bi \* bi1).Convert2Int()}");  
 Console.WriteLine($"{bi.Convert2Int()} / {bi1.Convert2Int()} = {(bi / bi1).Convert2Int()}");  
 Console.WriteLine($"{bi.Convert2Int()} % {bi1.Convert2Int()} = {(bi % bi1).Convert2Int()}");  
 Console.WriteLine($"{bi.Convert2Int()} == {bi1.Convert2Int()} = {(bi == bi1)}");  
 Console.WriteLine($"{bi.Convert2Int()} < {bi1.Convert2Int()} = {(bi < bi1)}");  
 Console.WriteLine($"{bi.Convert2Int()} > {bi1.Convert2Int()} = {(bi > bi1)}");  
 Console.WriteLine($"{bi.Convert2Int()} != {bi1.Convert2Int()} = {(bi != bi1)}");  
   
   
 }  
}

Вывод:  
  
456 + 874 = 1330

456 - 874 = -418

456 \* 874 = 398544

456 / 874 = 0

456 % 874 = 456

456 == 874 = False

456 < 874 = True

456 > 874 = False

456 != 874 = True

Тестирование RSA:

namespace Lab1;  
  
public class Main1  
{  
 private static void Main()  
 {  
 *// Пример работы RSA шифровальщика/дешифровальщика:* var rsa = new RSA();  
 rsa.Encrypt();  
   
 *// Результат шифрования в файле "output.txt"  
 // Дешифруем!* rsa.Decrypt();  
   
 *// Результат дешифрования в "result.txt"* }  
}

Файл input.txt:   
 Привет, Мир

Файл output.txt:

147429

177822

82085

98502

176676

31405

85227

13342

125620

82085

177822  
  
Файл result.txt:  
 Привет, Мир

Файл keysDnN.txt:

180599

181451

# Вывод

Я сумел реализовать RSA шифрование. Программа сейчас работает долго, потому думаю, что есть намного более эффективное решение. В процессе выполнения я вспомнил про StreamReader’ы и StreamWriter’ы, что очень мне помогло. Самым сложным для меня было понять суть RSA шифрования.