

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГУ»)

Факультет прикладной математики информатики и механики  
Кафедра ERP-систем и бизнес-процессов

**Криптосистема Эль-Гамала.  
Лабораторная работа**

Магистерская диссертация  
Направление 01.04.02 прикладная математика и информатика  
Магистерская программа Математическое моделирование

Допущено к защите в ГЭК 31 мая 2018 года

Зав. Кафедрой \_\_\_\_\_  
Обучающийся \_\_\_\_\_  
Руководитель \_\_\_\_\_

Йорг Беккер  
В. А. Ковун  
к.т.н. доцент Б. Н. Воронков

## СОДЕРЖАНИЕ

Введение . . . . .	3
Постановка задачи . . . . .	4
1. Общая информация о криптосистеме Эль-Гамала . . . . .	5
1.1 Алгоритм создания открытого и закрытого ключей . . . . .	6
1.2. Шифрование и расшифрование . . . . .	6
1.3. Дешифрование . . . . .	7
1.4. Особенности криптосистемы Эль-Гамала . . . . .	7
2. Алгоритмы решения задачи дискретного логарифмирования . . . . .	9
2.1. В произвольной мультипликативной группе . . . . .	9
2.2. В кольце вычетов по простому модулю . . . . .	9
2.3. Алгоритмы с экспоненциальной сложностью . . . . .	10
2.4. Субэкспоненциальные алгоритмы . . . . .	12
3. Американский стандарт кодирования - ASCII . . . . .	14
4. Анализ DES, ГОСТ 28147-89, Crypto03, El-Gamal . . . . .	16
5. Описание электронной обучающей программы "El-Gamal_Tutor" . . . . .	20
5.1 Общие сведения . . . . .	20
5.2. Функциональное назначение . . . . .	20
5.3. Используемые технические средства . . . . .	20
5.4. Описание логической структуры . . . . .	21
5.5. Описание алгоритма . . . . .	22
5.6. Вызов и загрузка . . . . .	48
5.7. Входные и выходные данные . . . . .	48
6. Описание сценария лабораторной работы . . . . .	49
6.1. Постановка задачи . . . . .	49
6.2. Содержание отчета о выполнении лабораторной работы . . . . .	49
Заключение . . . . .	51
Список литературы . . . . .	52
Приложение . . . . .	54

## ВВЕДЕНИЕ

В настоящее время в вузах Российской Федерации существующие базовые стандарты обучения для ряда специальностей включают в себя разделы, связанные с изучением методов и средств защиты информации. Для успешного освоения данных тем необходимо понимание принципов и знание основных элементов криптографического преобразования информации.

В Интернете можно найти десятки описаний лабораторных работ, посвященных криптографической системе Эль Гамала [1 – 3]. К сожалению, подавляющее большинство из них содержат задания и примеры реализации схемы Эль Гамала без учета особенностей длинной арифметики, не требуя обоснований алгоритмов и использования обучающих программ, не затрагивая вопросы криптоанализа.

Известно несколько компьютерных обучающих программ, позволяющих быстро и достаточно полно ознакомиться с алгоритмами шифрования и расшифрования данных, используемыми в традиционных симметричных и современных асимметричных криптосистемах. К сожалению, эти программы, представленные в сети Интернет, не сопровождаются исходными текстами, ограничиваются краткой справочной информацией и содержат большое число ошибок и недочетов. В связи с этим и было принято решение: разработать алгоритм и реализовать свою электронную обучающую программу для изучения криптосистемы Эль Гамала, а также разработать сценарий лабораторной работы с использованием этой программы. Предлагаемый вариант лабораторной работы призван преодолеть указанные недостатки.

## ПОСТАНОВКА ЗАДАЧИ

1. Провести анализ криптографического алгоритма Эль Гамала.
2. Разработать сценарий выполнения лабораторной работы по изучению алгоритма Эль Гамала.
3. Ознакомиться с обучающими программами по криптографии: DES, ГОСТ 28147-89, Crypto-03, Elgamal, выявить их достоинства и недостатки.
4. Разработать и реализовать обучающую компьютерную программу "El-Gamal\_Tutor".

# 1. ОБЩАЯ ИНФОРМАЦИЯ О КРИПТОСИСТЕМЕ ЭЛЬ-ГАМАЛЯ

Схема Эль-Гамала (Elgamal) — криптографическая система с открытым ключом, стойкость которой основана на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя как алгоритм шифрования, так и алгоритм цифровой подписи. Криптосистема Эль-Гамала лежит в основе бывших стандартов электронной цифровой подписи в США (DSA) и России (ГОСТ Р 34.10-94, ГОСТ Р 34.10-2001). Криптосистема была предложена египетским математиком и криптографом Тахером Эль-Гамалем в 1985 году. Тахер Эль-Гамаль разрабатывал один из вариантов алгоритма Диффи-Хеллмана. Он усовершенствовал систему Диффи-Хеллмана и получил два алгоритма, которые использовались для шифрования и для обеспечения аутентификации. В отличие от RSA, алгоритм Эль-Гамала не был запатентован, и поэтому стал более дешевой альтернативой, так как при использовании криптосистемы не требовалась оплата взносов за лицензию. Считается, что алгоритм попадает под действие патента Диффи-Хеллмана.

Криптографические системы с открытым ключом, также называемые асимметричными, используют так называемые односторонние функции, которые обладают следующим свойством:

- Если известен аргумент  $x$ , то  $f(x)$  вычислить относительно просто
- Если известно значение функции  $y = f(x)$ , то для вычисления аргумента  $x$  нет простого (вычислительно эффективного) пути.

Под односторонностью понимается не теоретическая однонаправленность функции, а практическая невозможность вычислить обратное значение за обозримый интервал времени, даже используя современные алгоритмы и вычислительные средства.

В основу криптографической системы Эль-Гамала положена сложность задачи дискретного логарифмирования в конечном поле. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования за разумное время необходимо уметь вычислять дискретный логарифм в конечном поле по простому модулю, что является вычислительно трудной задачей.

В криптографической системе с открытым ключом каждый участник располагает как открытым ключом (англ. public key), так и закрытым ключом (англ. private key). В криптографической системе Эль-Гамала открытый ключ состоит из тройки неотрицательных целых чисел, а закрытый ключ состоит из одного неотрицательного целого числа. Каждый участник создаёт свой открытый и закрытый ключ самостоятельно. Закрытый ключ каждый из них держит в секрете, а открытые ключи можно сообщать кому угодно, передавать по незащищённым каналам или даже публиковать их.

## 1.1. Алгоритм создания открытого и закрытого ключей

Ключи в схеме Эль-Гамала генерируются следующим образом:

1. Генерируется случайное простое число  $p$ .
2. Выбирается целое число  $g$  — первообразный корень  $p$ .
3. Выбирается случайное целое число  $x$ , такое, что  $1 < x < p$ .
4. Вычисляется  $y = g^x \bmod p$ .
5. Открытым ключом является тройка  $(p, g, y)$ , закрытым ключом — число  $x$ .

## 1.2. Шифрование и расшифрование

Предположим, пользователь А намеревается послать пользователю Б некоторое сообщение. Сообщениями являются целые неотрицательные числа в интервале от 0 до  $p - 1$ . Алгоритм для шифрования:

1. Взять открытый ключ пользователя Б.
2. Взять открытый текст  $M$ .
3. Выбрать сессионный ключ — случайное целое число  $k$  такое, что  $1 < k < p - 1$ .
4. Зашифровать сообщение с использованием открытого ключа пользователя Б, то есть вычислить числа:  $a = g^k \bmod p$ , и  $b = y^k M \bmod p$ .

Алгоритм для расшифрования:

1. принять зашифрованное сообщение  $(a, b)$  от пользователя А.
2. Взять свой закрытый ключ  $M$ .
3. Применить закрытый ключ для расшифрования сообщения:  $M = b(a^x)^{-1} \bmod p$
4. При этом нетрудно проверить, что  $(a^x)^{-1} \equiv g^{-kx} \pmod{p}$ , и поэтому  $b(a^x)^{-1} \equiv (y^k M)g^{-xk} \equiv (g^{xk} M)g^{-xk} \equiv M \pmod{p}$ .

### 1.3. Дешифрование

Дешифрование - получение открытых данных по зашифрованным в условиях, когда алгоритм расшифрования и его секретные параметры не являются полностью известными и расшифрование не может быть выполнено обычным путем. Алгоритм для дешифрования криптосистемы Эль-Гамала:

1. Перехватить зашифрованное сообщение  $(a, b)$ .
2. Взять открытый ключ  $(p, g, y)$ .
3. Решить относительно  $x$  уравнение  $y \equiv g^x \pmod{p}$ .
4. Расшифровать сообщение по формуле  $M = b(a^x)^{-1} \bmod p$ .

Собственно, самый главный вопрос из этого алгоритма – как по данным  $(p, g, y)$  найти  $x$ . Эта задача называется задачей дискретного логарифмирования [2].

### 1.4. Особенности криптосистемы Эль-Гамала

- Криптосистема асимметричная (двухключевая).
- Блочная, с длиной блока открытого текста меньше или равной длине открытого (публичного) ключа.
- Длина открытого и закрытого ключей, по современным представлениям, 2048 бит или более.

- Используется лишь один метод шифрования – метод аналитических преобразований.
- Базируется на вычислительно трудной задаче дискретного логарифмирования.
- Предоставляет возможность реализации электронной подписи.



## 2. АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ ДИСКРЕТНОГО ЛОГАРИФМИРОВАНИЯ

### 2.1. В произвольной мультипликативной группе

Разрешимости и решению задачи дискретного логарифмирования в произвольной конечной абелевой группе посвящена статья J. Buchmann, M. J. Jacobson и E. Teske [9]. В алгоритме используется таблица, состоящая из  $O(\sqrt{|g|})$  пар элементов, и выполняется  $O(\sqrt{|g|})$  умножений. Данный алгоритм медленный и не пригоден для практического использования. Для конкретных групп существуют свои, более эффективные, алгоритмы.

### 2.2. В кольце вычетов по простому модулю

Рассмотрим сравнение

$$a^x \equiv b \pmod{p} \quad (1)$$

где  $p$  — простое,  $b$  не делится на  $p$ . Если  $a$  является образующим элементом группы  $\mathbb{Z}/p\mathbb{Z}$ , то сравнение (1) имеет решение при любых  $b$ . Такие числа  $a$  называются ещё первообразными корнями, и их количество равно  $\phi(p) = p - 1$ , где  $\phi$  — функция Эйлера. Решение сравнения (1) можно находить по формуле:

$$x \equiv \sum_{i=1}^{p-2} (1 - a^i)^{-1} b^i \pmod{p} \quad (2)$$

Однако, сложность вычисления по этой формуле хуже, чем сложность полного перебора.

Следующий алгоритм [3] имеет сложность  $O(\sqrt{p} \cdot \log p)$ . Алгоритм

1. Присвоить  $H := \lfloor \sqrt{p} \rfloor + 1$
2. Вычислить  $c = a^H \pmod{p}$
3. Составить таблицу значений  $c^u \pmod{p}$  для  $1 \leq u \leq H$  и отсортировать её.

4. Составить таблицу значений  $b \cdot a^v \bmod p$  для  $0 \leq v \leq H$  и отсортировать её.
5. Найти общие элементы в таблицах. Для них  $c^u \equiv b \cdot a^v \pmod{p}$  откуда  $a^{H \cdot u - v} \equiv b \pmod{p}$
6. Выдать  $H \cdot u - v$ .

Существует также множество других алгоритмов для решения задачи дискретного логарифмирования в поле вычетов [3]. Их принято разделять на экспоненциальные и субэкспоненциальные. Полиномиального алгоритма для решения этой задачи пока не найдено.

### 2.3. Алгоритмы с экспоненциальной сложностью

Алгоритм Гельфонда-Шенкса (алгоритм больших и малых шагов, baby-step giant-step) был предложен независимо советским математиком Александром Гельфондом в 1962 году и Дэниэлем Шенксом в 1972 году. Относится к методам встречи посередине.

Идея алгоритма состоит в выборе оптимального соотношения времени и памяти, а именно в усовершенствованном поиске показателя степени.

Пусть задана циклическая группа  $G$  порядка  $n$ , генератор группы  $\alpha$  и некоторый элемент группы  $\beta$ . Задача сводится к нахождению целого числа  $x$ , для которого выполняется  $\alpha^x = \beta \bmod m$ .

Алгоритм Гельфонда — Шенкса основан на представлении  $x$  в виде  $x = i \cdot t - j$ , где  $t = \lfloor \sqrt{n} \rfloor + 1$ , и переборе  $1 \leq i \leq t$  и  $0 \leq j \leq t$ . Ограничение на  $i$  и  $j$  следует из того, что порядок группы не превосходит  $t$ , а значит указанные диапазоны достаточны для получения всех возможных из полуинтервала  $[0; t)$ . Такое представление равносильно равенству

$$\alpha^{im} = \beta \alpha^j \tag{3}$$

Алгоритм предварительно вычисляет  $\alpha^{im}$  для разных значений  $i$  и сохраняет их в структуре данных, позволяющей эффективный поиск, а затем перебирает всевозможные значения  $j$  и проверяет, если  $\beta \alpha^j$  соответствует какому-то значению  $i$ . Таким образом находятся индексы  $i$  и  $j$ , которые удовлетворяют соотношению (3) и позволяют вычислить значение  $x = i \cdot t - j$ .

Алгоритму Гельфонда — Шенкса требуется  $O(n)$  памяти, что является недостатком данного метода. Возможно выбрать меньшее  $m$  на первом шаге алгоритма, но это увеличивает время работы программы до  $O(n/m)$ .

Другим методом дискретного логарифмирования является алгоритм Сильвера-Полига-Хеллмана. Он работает, если известно разложение числа  $p - 1 = \prod_{i=1}^s q_i^{\alpha_i}$  на простые множители. Сложность оценивается как  $O(\sum_{i=1}^s \alpha_i (\log p + q_i))$ . Если множители, на которые раскладывается  $p - 1$ , достаточно маленькие (другими словами, если  $p - 1$  - гладкое число), то алгоритм чрезвычайно эффективен. Это необходимо учитывать в выборе параметров при разработке криптографических схем, основанных на вычислительной сложности дискретного логарифмирования, иначе схема будет ненадёжной.

Для применения алгоритма Сильвера-Полига-Хеллмана необходимо знать разложение  $p - 1$  на простые множители. В общем случае задача факторизации — достаточно трудоёмкая, однако если делители числа — небольшие, то это число можно быстро разложить на множители даже методом последовательного деления. Таким образом, в тех случаях, когда эффективен алгоритм Сильвера-Полига-Хеллмана, необходимость факторизации не усложняет задачу в терминах вычислительной сложности.

Ещё одним методом дискретного логарифмирования является  $\rho$ -метод Полларда, который был предложен Джоном Поллардом в 1978 году, основные идеи алгоритма похожи на  $\rho$ -алгоритм Полларда для факторизации чисел. Условием работы  $\rho$ -метода Полларда является простота порядка группы, порождённой основанием  $a$  дискретного логарифма по модулю  $p$ .

Алгоритм имеет эвристическую оценку сложности  $O(p^{\frac{1}{2}})$ . По сравнению с другими методами дискретного логарифмирования  $\rho$ -метод Полларда является менее затратным как по отношению к вычислительным операциям, так и по отношению к затрачиваемой памяти. Например, при достаточно больших значениях числа  $p$  данный алгоритм является вычислительно менее сложным, чем алгоритм COS и алгоритм Адлемана. С другой стороны, условие обязательной простоты порядка мультипликативной группы для работы алгоритма накладывает серьёзные ограничения на его использование.

## 2.4. Субэкспоненциальные алгоритмы

Данные алгоритмы имеют сложность, оцениваемую как  $O(\exp(c(\log p \log p \log p)^d))$  арифметических операций, где  $c$  и  $0 \leq d \leq 1$  — некоторые константы. Эффективность алгоритма во многом зависит от близости константы  $c$  к 1 и константы  $d$  — к 0.

Алгоритм Адлемана [10] появился в 1979 году. Это был первый предложенный субэкспоненциальный алгоритм дискретного логарифмирования. На практике он всё же недостаточно эффективен. В этом алгоритме  $d = \frac{1}{2}$ .

Алгоритм COS [3] был предложен в 1986 году математиками Копперсмитом (Don Coppersmith), Одлышко (Andrew Odlyzko) и Шреппелем (Richard Schroepel). В этом алгоритме константа  $c = 1$ ,  $d = \frac{1}{2}$ . В 1991 году с помощью этого метода было проведено логарифмирование по модулю  $p \approx 10^{58}$ . В 1997 году Вебер [3] провел дискретное логарифмирование по модулю  $p \approx 10^{85}$  с помощью некоторой модифицированной версии данного алгоритма. Экспериментально показано, что при  $p \leq 10^{90}$  алгоритм COS эффективнее решета числового поля.

Дискретное логарифмирование при помощи решета числового поля [3] было применено к дискретному логарифмированию позднее, чем к факторизации чисел. Первые идеи появились в 1990-х годах. Алгоритм, предложенный Д. Гордоном в 1993 году [3], имел эвристическую сложность  $O(\exp 3^{3/2}(\log p \log p \log p)^{\frac{1}{3}})$ , но оказался достаточно непрактичным. Позднее было предложено множество различных улучшений данного алгоритма. Было показано, что при  $p \geq 10^{100}$  решето числового поля быстрее, чем COS [3]. Современные рекорды в области дискретного логарифмирования достигнуты именно с помощью этого метода.

Наилучшими параметрами в оценке сложности на данный момент является  $c = (92 + 26\sqrt{13})^{1/3}/3 \approx 1,902$ ,  $d = \frac{1}{3}$ . Для чисел специального вида результат можно улучшить. В некоторых случаях можно построить алгоритм, для которого константы будут  $c \approx 1,00475$ ,  $d = \frac{2}{5}$ . За счёт того, что константа  $c$  достаточно близка к 1, подобные алгоритмы могут обогнать алгоритм с  $d = \frac{1}{3}$ .

Другая возможность эффективного решения задачи вычисления дискретного логарифма связана с квантовыми вычислениями. Теоретически доказано, что с их помощью можно произвести вычисление дискретного

логарифма за полиномиальное время. В любом случае, если полиномиальный алгоритм вычисления дискретного логарифма будет реализован, это будет означать практически полную непригодность криптосистем на его основе [3].



### 3. АМЕРИКАНСКИЙ СТАНДАРТ КОДИРОВАНИЯ - ASCII

ASCII (англ. American Standard Code for Information Interchange) — американская стандартная 7-битная кодировочная таблица для печатных символов и некоторых специальных кодов, используемая в компьютерной коммуникации. ASCII представляет собой кодировку для представления десятичных цифр, латинского и национального алфавитов, знаков препинания и управляющих символов.

Таблица была разработана и стандартизована в 1963 году. Множество современных кодировок и стандартов (UTF-8, Win-1251, КОИ-8) являются расширениями стандарта ASCII. В СССР стандарт был утвержден в 1987 году в виде таблицы международной ссылочной версии кода КОИ-7 НО ГОСТ 27463-87 (СТ СЭВ 356-86) «Системы обработки информации. 7-битные кодированные наборы символов» [4].

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	\$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

Рис. 1. ASCII коды

В криптографических программах ASCII используется для преобразования символов текста в цифры, чтобы текст было возможно представить в виде чисел и совершать над ним криптографические

преобразования. Например: большим буквам английского алфавита соответствуют значения с 97 по 122 (рис. 1).

Поскольку на подавляющем большинстве современных компьютеров минимально адресуемой единицей памяти является байт (размером в 8 бит), там используются 8-битные, а не 7-битные символы. Обычно символ ASCII расширяют до 8 бит, подставляя нулевой бит в качестве старшего. Таким образом, каждый преобразованный в число символ занимает ровно один байт. Уменьшение размера одного символа для криптосистем главным образом означает возможность передать большой шифротекст в одном блоке при неизменной длине ключа.

## 4. АНАЛИЗ DES, ГОСТ 28147-89, CRYPTO03, EL-GAMAL

Перед началом написания программы “El-Gamal\_Tutor” были изучены другие приложения для обучения криптосистемам. Одними из них были: DES, ГОСТ 28147-89, Crypto-03 и El-Gamal.

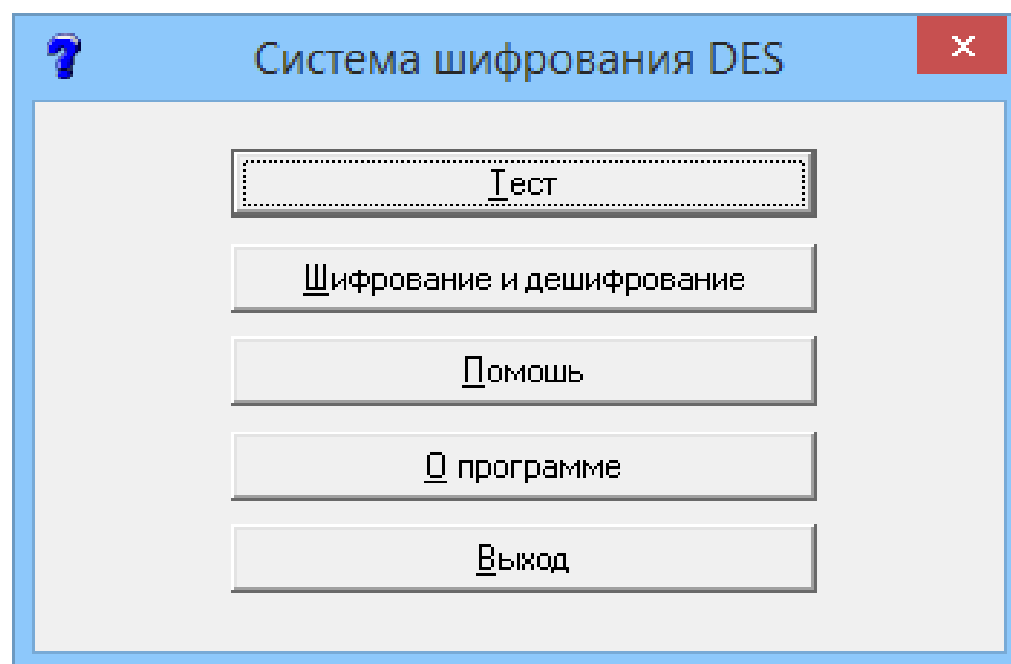


Рис. 2. Главное меню программы "Система шифрования DES"

Программа "Система шифрования DES" (рис. 2–4) предлагает режим обучения симметричной криптосистеме DES, а так же, в качестве дополнительной функции, возможность зашифровать и расшифровать произвольное сообщение используя криптосистему DES.

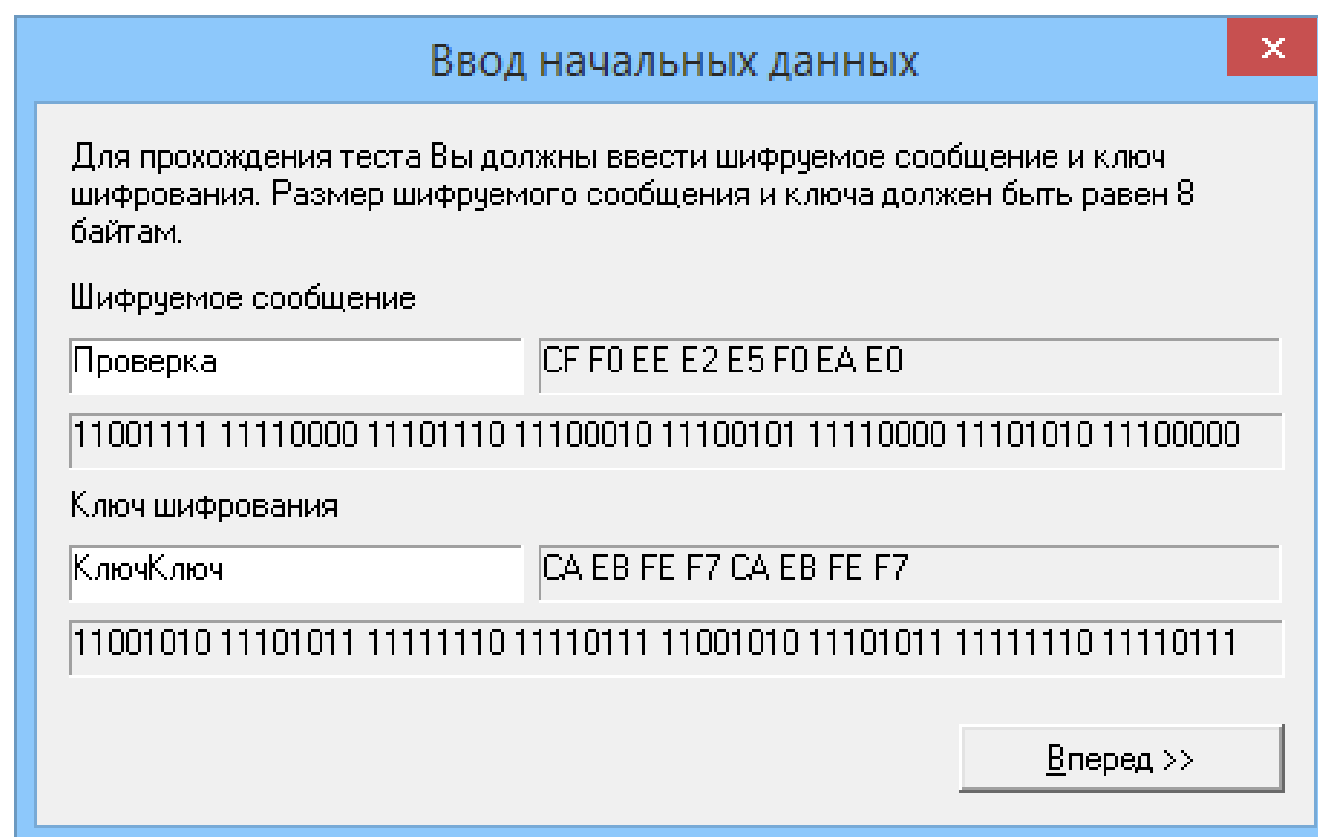


Рис. 3. Ввод начальных данных



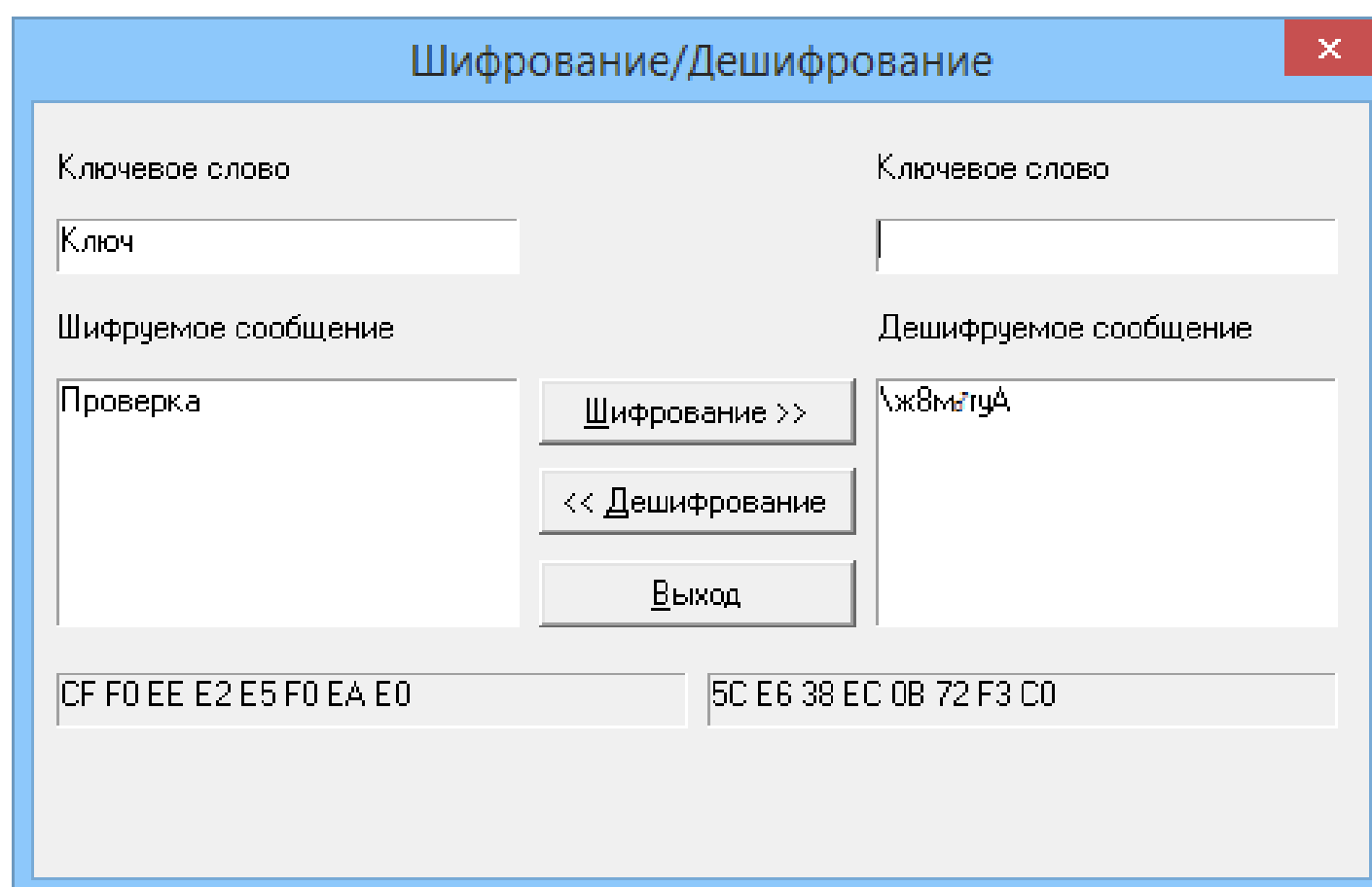


Рис. 4. Шифрование/Расшифрование [5]

В данном случае в программе режим называется неправильно, так как на самом деле вместо дешифрования происходит расшифрование.

К сожалению, программа не предлагает дополнительных возможностей, таких, как отдельный режим проведения криптографических вычислений и преобразований.

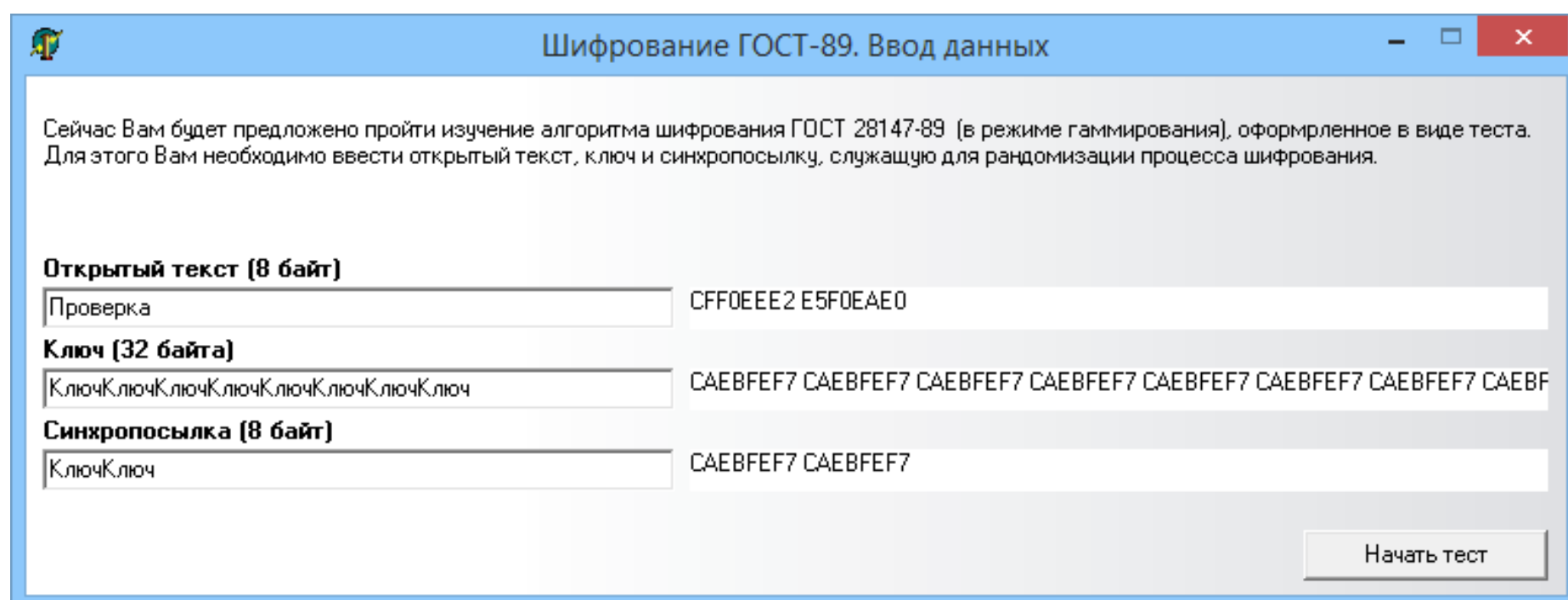


Рис. 5. Ввод данных в обучающей программе ГОСТ 28147-89

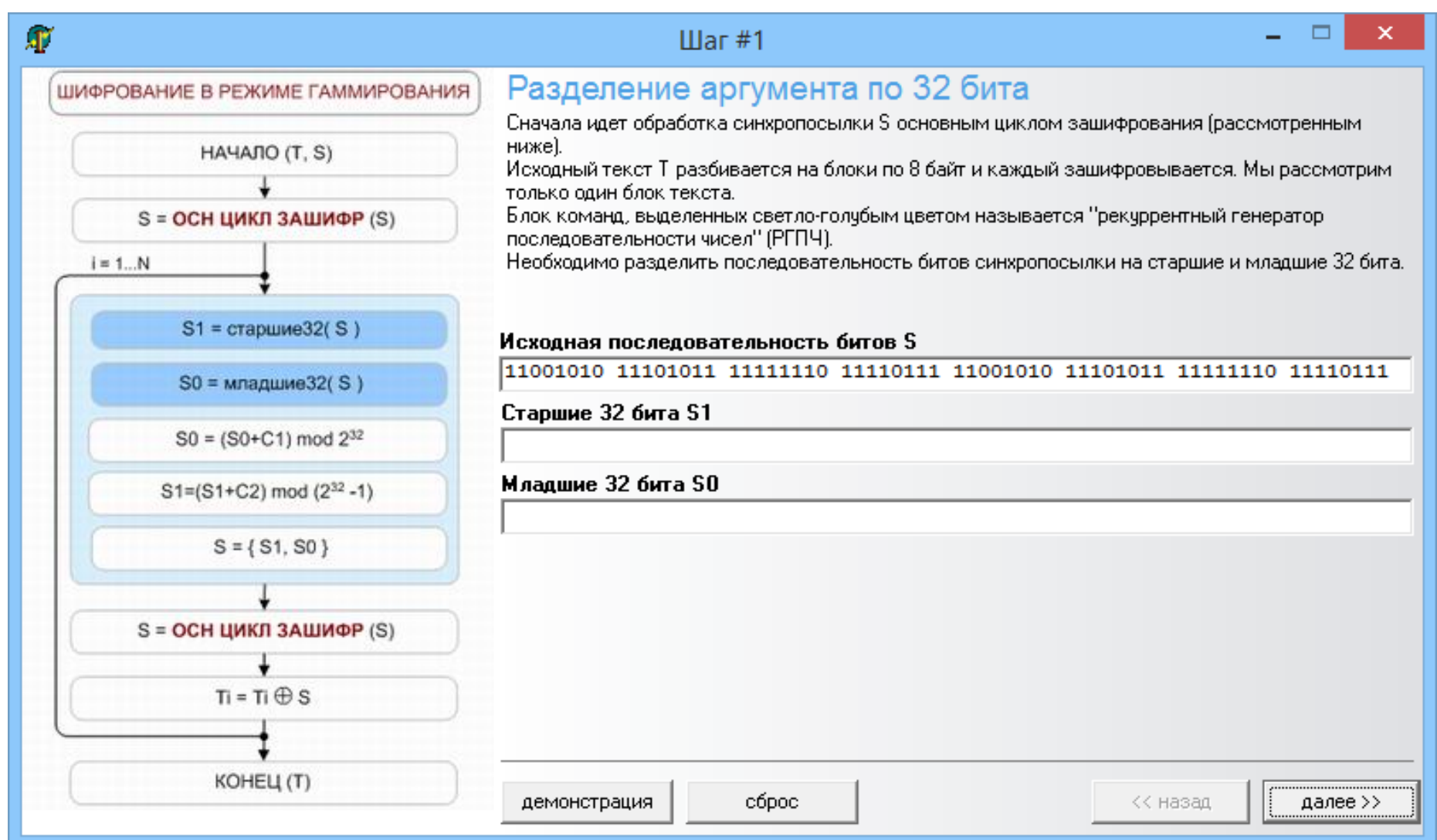


Рис. 6. Первый шаг обучения ГОСТ 28147-89 [7]

Как и программа "Система шифрования DES", программа ГОСТ 28147-89 (рис. 5, 6) не предлагает дополнительных криптографических или математических возможностей. Она также не предлагает возможности опробовать криптографический алгоритм ГОСТ-89 на произвольном сообщении без необходимости проходить при этом шаги обучения криптосистеме.

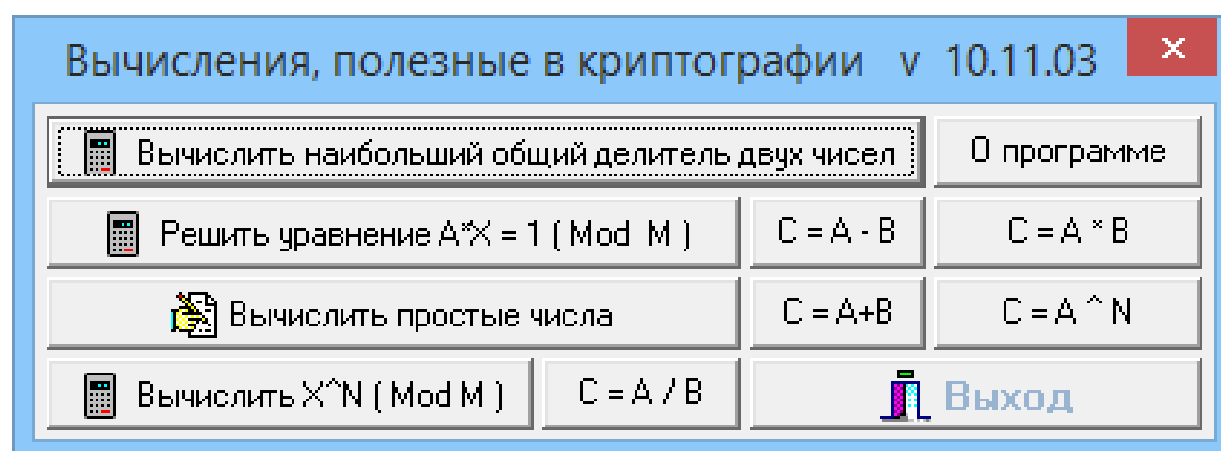


Рис. 7. Основная форма программы вычислений, полезных в криптографии v.10.11.2003 – Crypto03 [5]

Программа Crypto03 (рис. 7, 8) представляет собой своего рода криптографический калькулятор, содержащий в себе ряд вычислительных функций, полезных в криптографии. Она не предоставляет режима обучения.

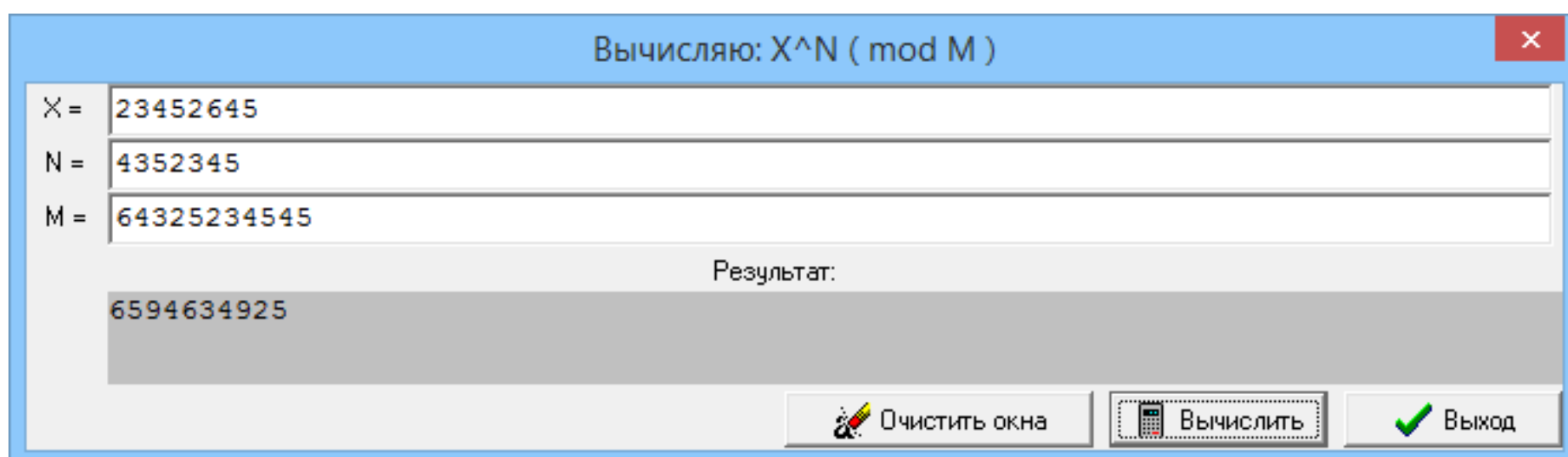


Рис. 8. Вычисление  $X^N \bmod M$  в программе Crypto03

Программа El-Gamal (рис. 9) - обучающая программа, посвящённая криптосистеме Эль-Гамала. Основными недостатками программы являются скудная подача обучающего материала и весьма неудобный интерфейс. Программа не предлагает дополнительных криптографических функций, а также проблематична в освоении без использования документации.

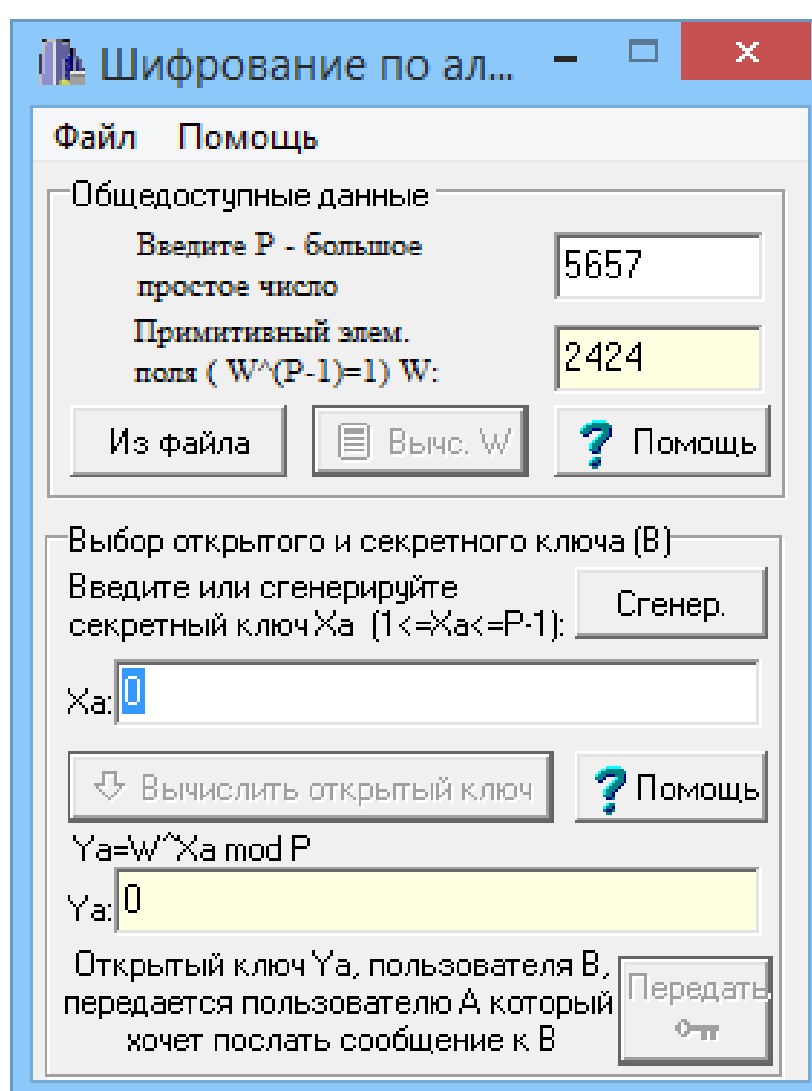


Рис. 9. Основная форма обучающей программы El-Gamal [6]

После рассмотрения всех этих программ, была сформирована картина того, как должна выглядеть будущая электронная обучающая программа El-Gamal\_Tutor.

## **5. ОПИСАНИЕ ЭЛЕКТРОННОЙ ОБУЧАЮЩЕЙ ПРОГРАММЫ "EL-GAMAL\_TUTOR"**

Посредством среды программирования Microsoft Visual Studio Community 2017 создано приложение, предназначенное для обучения основам криптографической системы Эль-Гамала.

### **5.1. Общие сведения**

Программа написана на языке программирования C# в визуальной среде Microsoft Visual Studio 2017 Community Edition с использованием программной платформы Microsoft .NET Framework 4.5. Проект общим объемом 1.84 Мб. Программа функционирует в операционной системе Windows 7 или новее.

При разработке использовались модули: System.Collections.Generic, System.ComponentModel, System.Data, System.Drawing, System.Linq, System.Text, System.Threading.Tasks, System.Windows.Forms, System.Numerics.

Размер генерируемых программой ключей теоретически ничем не ограничен, практически же он ограничен в соответствии с характеристиками компьютера, на котором запускается программа.

### **5.2. Функциональное назначение**

Приложение предназначено для обучения методам и алгоритмам, используемым при реализации асимметричной криптографической системы Эль-Гамала, а также частичной проверки знаний учащегося.

Дополнительные функции приложения позволяют использовать его в качестве программы для небольших полезных в криптографии вычислений.

### **5.3. Используемые технические средства**

Компьютер с шестиядерным процессором 3.2 GHz, 8 Gb RAM, Microsoft Windows 10 x64.

## 5.4. Описание логической структуры

Программа логически разделена на две части: режим обучения и вспомогательные функции. Блок-схема алгоритма работы приведена на рис. 10.

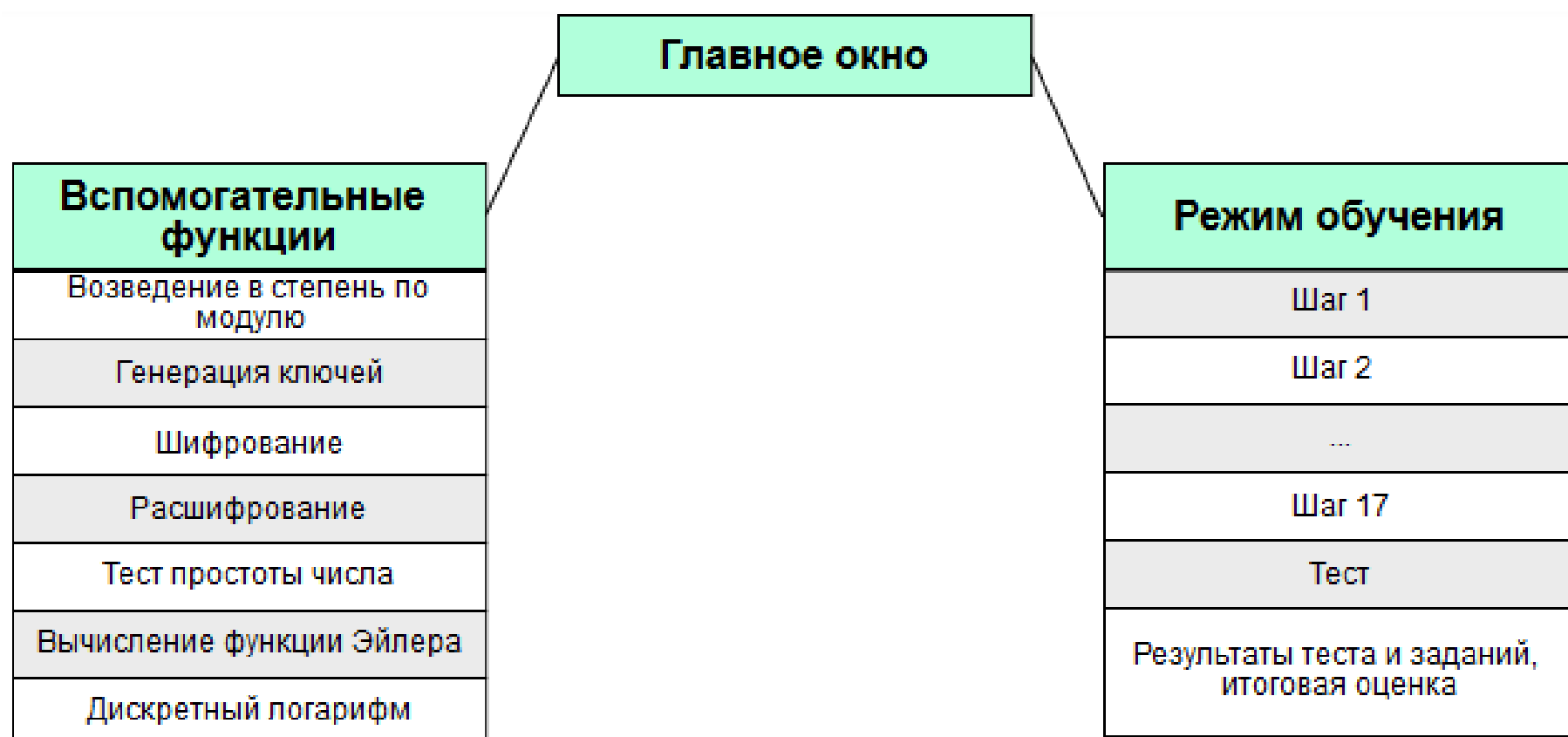


Рис. 10. Блок-схема программы "El-Gamal\_Tutor"

В режиме обучения рассматриваются математические основы, на которых базируется криптосистема Эль-Гамала, алгоритмы генерации ключей, шифрования и расшифрования, а также основы криптоанализа системы и некоторые алгоритмы дискретного логарифмирования. Дополнительный функционал включает в себя различные вычислительные возможности, так или иначе связанные с криптосистемой Эль-Гамала. Они могут использоваться как в совокупности с обучением криптосистеме, так и отдельно от него.

## 5.5. Описание алгоритма

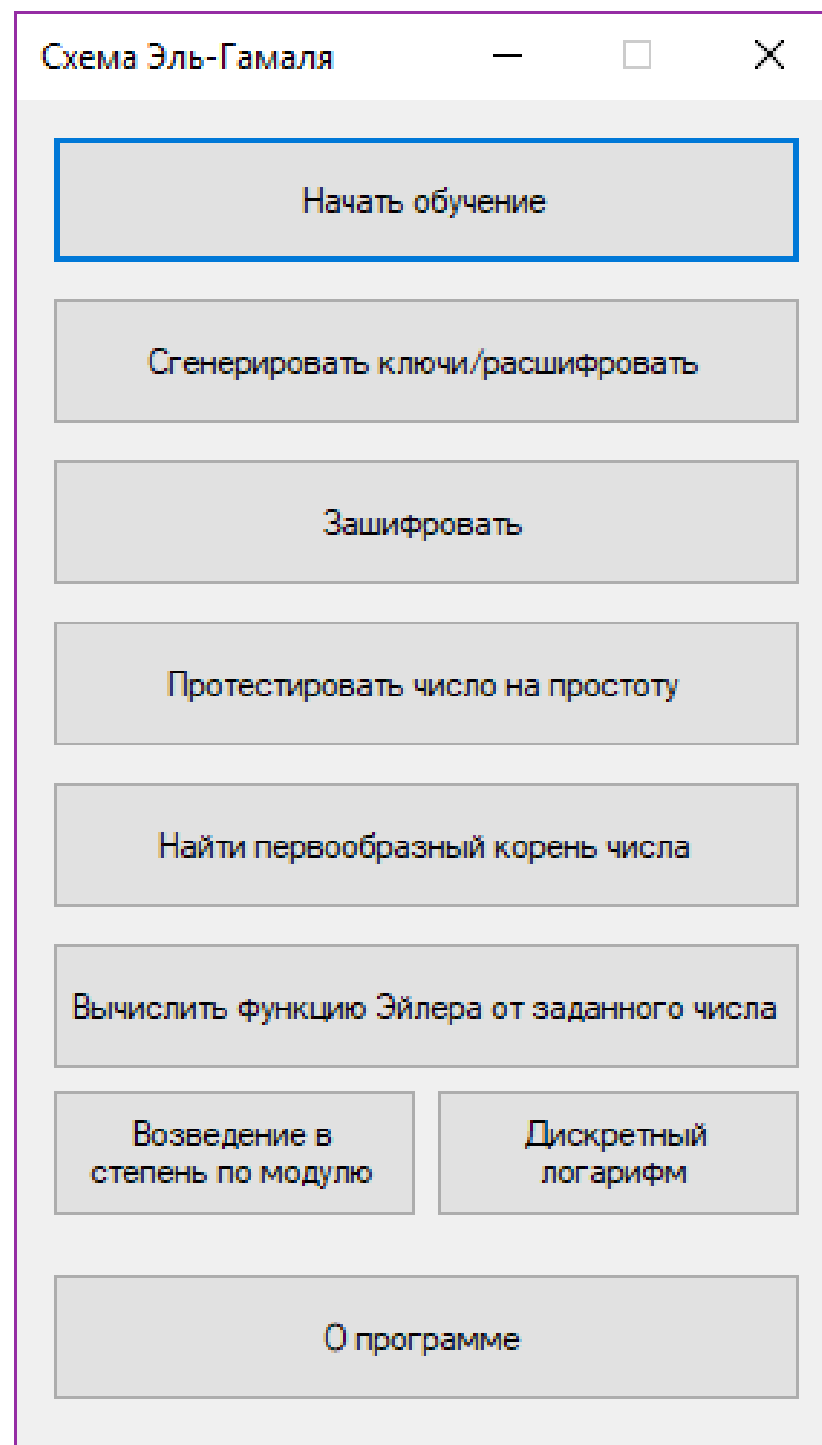


Рис. 11. Основное меню программы "El-Gamal\_Tutor"

На рис. 11 представлено основное меню программы. Вход в обучающий режим осуществляется нажатием кнопки "начать обучение".

Рис. 12. Генерация ключей и расшифрование

В режиме генерации ключей (рис. 12) мы можем сгенерировать ключи для криптосистемы Эль-Гамала, а также расшифровать необходимую фразу из шифротекста с использованием этих ключей.

Шифрование

Зашифрование:

$p =$  517467185476731163468517378827

$g =$  2

$y =$  308728593639302017025256895664

$k =$  1308825640321312561 Сгенерировать

Текст для зашифровки: Эль-Гамаль Шифровать!

Шифротекст:

$a =$  496976537142753426250778832233

$b =$  279270060747999426426084244771

Рис. 13. Результат шифрования

Режим шифрования, представленный на рис. 13, позволяет зашифровать сообщение пользователя с помощью введённого открытого ключа.

Тест простоты

Введите число:

2133361524376715562757651

Тест

Введённое число - простое

Рис. 14. Тест простоты произвольного числа

Режим теста простоты (рис. 14) позволяет проверить, является ли введённое целое неотрицательное число простым или составным. Для определения простоты числа в программе используется вероятностный тест Миллера-Рабина, количество «свидетелей простоты» - 4000.



Вычисление первообразного корня

Введите или сгенерируйте модуль: 18113443457355578543

кол-во разрядов: 20

Сгенерировать

Вычислить:

Полный метод: 5

Упрощённый метод: 5

Рис. 15. Вычисление первообразного корня по заданному модулю

Режим вычисления первообразного корня (рис. 15) позволяет вычислить первообразный (или примитивный) корень для большого числа. Поскольку полный метод вычисления первообразного корня очень медленен для больших чисел, в программе предусмотрена возможность вычисления первообразного корня по «упрощённому» методу, который даёт ответ, верный только с некоторой вероятностью.

Вычисление функции ...

Введите число: 150

Вычислить

Количество чисел, взаимно простых с введённым: 40

Рис. 16. Вычисление функции Эйлера

Режим вычисления функции Эйлера (рис. 16) позволяет вычислить количество чисел, взаимно простых с заданным.

$X^Y \text{ MOD } M$

$X =$  4345756111154649614113243787547487584525164216537582761

$Y =$  1532155624186748145651111563454156541385447544711623418

$M =$  5341265134211651344613486787819912316354487871654646726

Вычислить

$X^Y \text{ MOD } M =$  2899615178108335402708419232116921097863094883771643977

Рис. 17. Возведение в степень по модулю

Режим возведения в степень по модулю (рис. 17) представляет собой калькулятор заданных степеней произвольных чисел по заданному модулю.

Дискретный логарифм

$A^x = B \text{ mod } M$

Случайные данные

$A$  489089

$B$  25430788

$M$  555126753227

Алгоритм Гельфонда-Шенкса
 Р-метод Полларда
 Алгоритм Полига-Хеллмана

$X = 552147215997$

Рис. 18. Дискретное логарифмирование

Режим дискретного логарифмирования (рис. 18) позволяет произвести поиск решения уравнения  $A^X = B \text{ mod } M$  для произвольных целых чисел  $A$  и  $B$  и простого числа  $M$ . Для поиска решения пользователю предлагается использовать три алгоритма дискретного логарифмирования: алгоритм Гельфонда-Шенкса,  $\rho$ -метод Полларда и алгоритм Полига-Хеллмана.

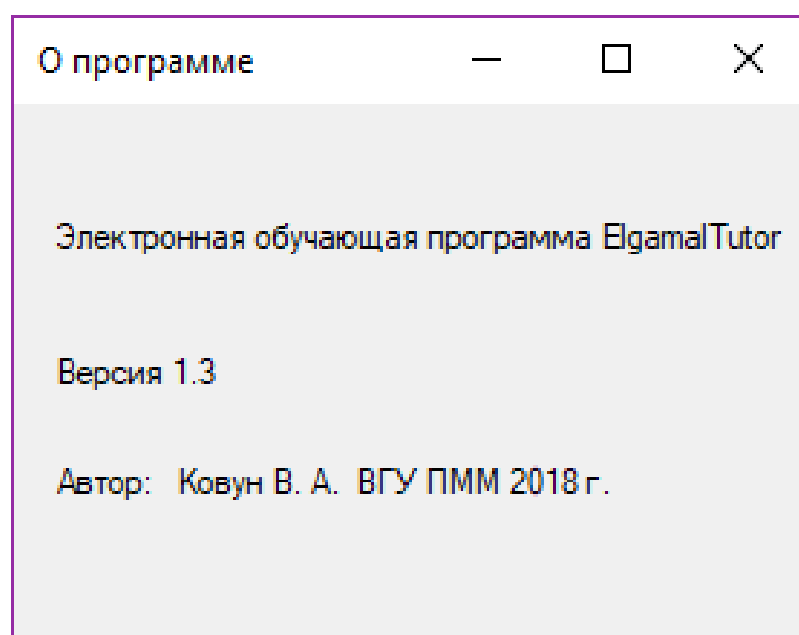


Рис. 19. О программе

В окне «О программе» (рис. 19) мы можем увидеть информацию о приложении El-Gamal\_Tutor.

Теперь перейдем к режиму обучения. Он состоит из нескольких шагов.

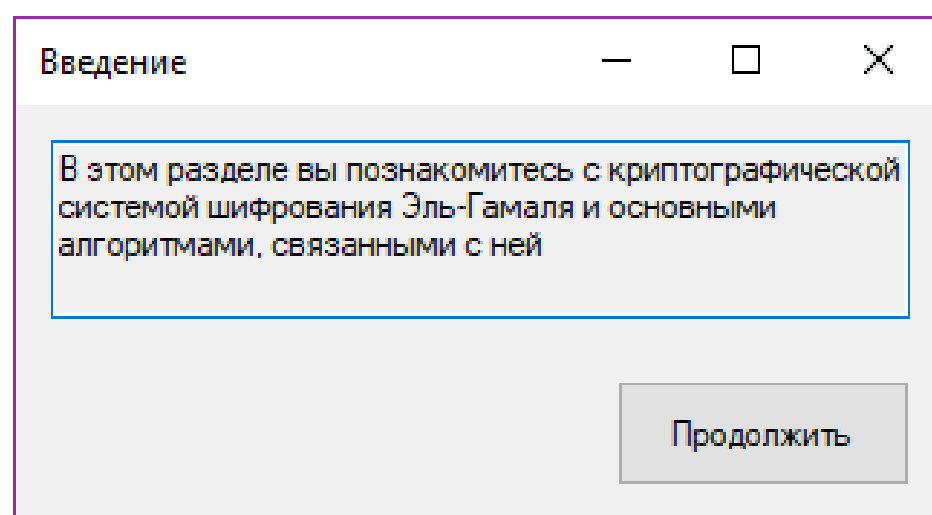


Рис. 20. Введение

На первом шаге (рис. 21) рассказывается про операцию возведения в степень по модулю и предлагается решить три примера. Условия заданий генерируются случайным образом.

Возведение в степень по модулю

— □ ×

Возведение в степень по модулю – это вычисление остатка от деления натурального числа  $b$  (основание), возведенного в степень  $e$  (показатель степени), на натуральное число  $m$  (модуль).

Например, пусть нам даны  $b = 5$ ,  $e = 3$  и  $m = 13$ , тогда решение  $s = 8$  - это остаток от деления  $5^3$  на 13.

Обозначение:  $s = b^e \bmod m$ .

Попробуйте возвести 3 в степень 3 по модулю 15

Ответ:

$9^4 \bmod 19 =$

Ответ:

$6^4 \bmod 13 =$

Ответ:

Далее

Рис. 21. Возведение в степень по модулю

На втором шаге (рис. 22) рассказывается про функцию Эйлера и предлагается решить три примера. Условия заданий так же генерируются случайным образом.

Функция Эйлера

Функция Эйлера  $\varphi(n)$  – мультипликативная арифметическая функция, равная количеству натуральных чисел, меньших  $n$  и взаимно простых с ним. При этом полагают, что число 1 взаимно просто со всеми натуральными числами, и  $\varphi(1)=1$ .

Например, для числа 24 существует 8 меньших него и взаимно простых с ним чисел (1, 5, 7, 11, 13, 17, 19, 23), поэтому  $\varphi(24)=8$ .

Для произвольного натурального числа  $n$  функция Эйлера может быть вычислена по следующей формуле, где  $p[1]...p[n]$  – простые числа, являющиеся делителями числа  $n$  согласно основной теореме арифметики:

$$\varphi\left(\prod_{i=1}^n p_i^{k_i}\right) = \prod_{i=1}^n (p_i^{k_i} - p_i^{k_i-1})$$

$\varphi(24) =$

$\varphi(9) =$

$\varphi(8) =$

Назад

Далее

Рис. 22. Функция Эйлера

На третьем шаге (рис. 23) объясняется операция нахождения обратного по модулю числа, и предлагается найти два таких числа для сгенерированных условий.

Нахождение обратного по модулю

В обычной арифметике  $a^{-1} = 1/a$ ,  $a \cdot (a^{-1}) = 1$ ,  $a \neq 0$ .  
 В модулярной арифметике  $x$  называется величиной, обратной  $a$  по модулю  $m$ , если выполняется сравнение  $a \cdot x = 1 \pmod m$ , при этом  $(a, m) = 1$  (т.е.  $a$  и  $m$  взаимно просты).  
 Основные способы нахождения обратных по модулю величин:  
 1. Подставляя поочередно вместо  $x$  значения  $1, 2, \dots, (m-1)$ , найти решение уравнения  $(a \cdot x) \pmod m = 1$

$x = 4^{-1} \pmod 9 =$

2. Если известна функция Эйлера  $\phi(m)$ , то  $(a^{-1}) \pmod m = a^{\phi(m)-1} \pmod m$ .

$x = 18^{-1} \pmod 29 =$

Рис. 23. Обратное по модулю число

На четвёртом шаге (рис. 24) пользователю рассказывается о египетском математике и криптографе Тахере Эль-Гамале, авторе рассматриваемой криптосистемы.

Тахер Эль-Гамаль

Египетский криптограф Тахер Эль-Гамаль родился в 1955 году. Свои первые шаги в криптографии он сделал путем вывода алгоритма Дискретных Квадратичных Логарифмов, который используется для факторизации целых чисел. Стал известен всему миру благодаря разработке «цифровой подписи по схеме Эль-Гамала» в 1984 году, после этого занимался в основном графикой и разработкой технологии сжатия данных в компании Hewlett-Packard. В 1985 году представил свои разработки по созданию систем асимметричного шифрования и цифровой подписи, эксплуатирующих сложность проблемы дискретного логарифмирования. Предложенная им схема ЭЦП стала основой для алгоритма DSA, принятого Национальным институтом стандартов и технологий США (NIST) в качестве стандарта цифровой подписи. В 1994 году стал активно развивать алгоритм SSL; результатом его работы было заключение договора с Microsoft, что сыграло одну из ключевых ролей в успехе SSL. Тахер Эль-Гамаль также участвовал в создании протокола оплаты по кредитной карте SET, а также ряда схем интернет-платежей.




Рис. 24. Тахер Эль-Гамаль

Пятый шаг (рис. 25) рассказывает общую информацию о схеме Эль-Гамала и основных стандартах где она использовалась ранее или используется по сей день.

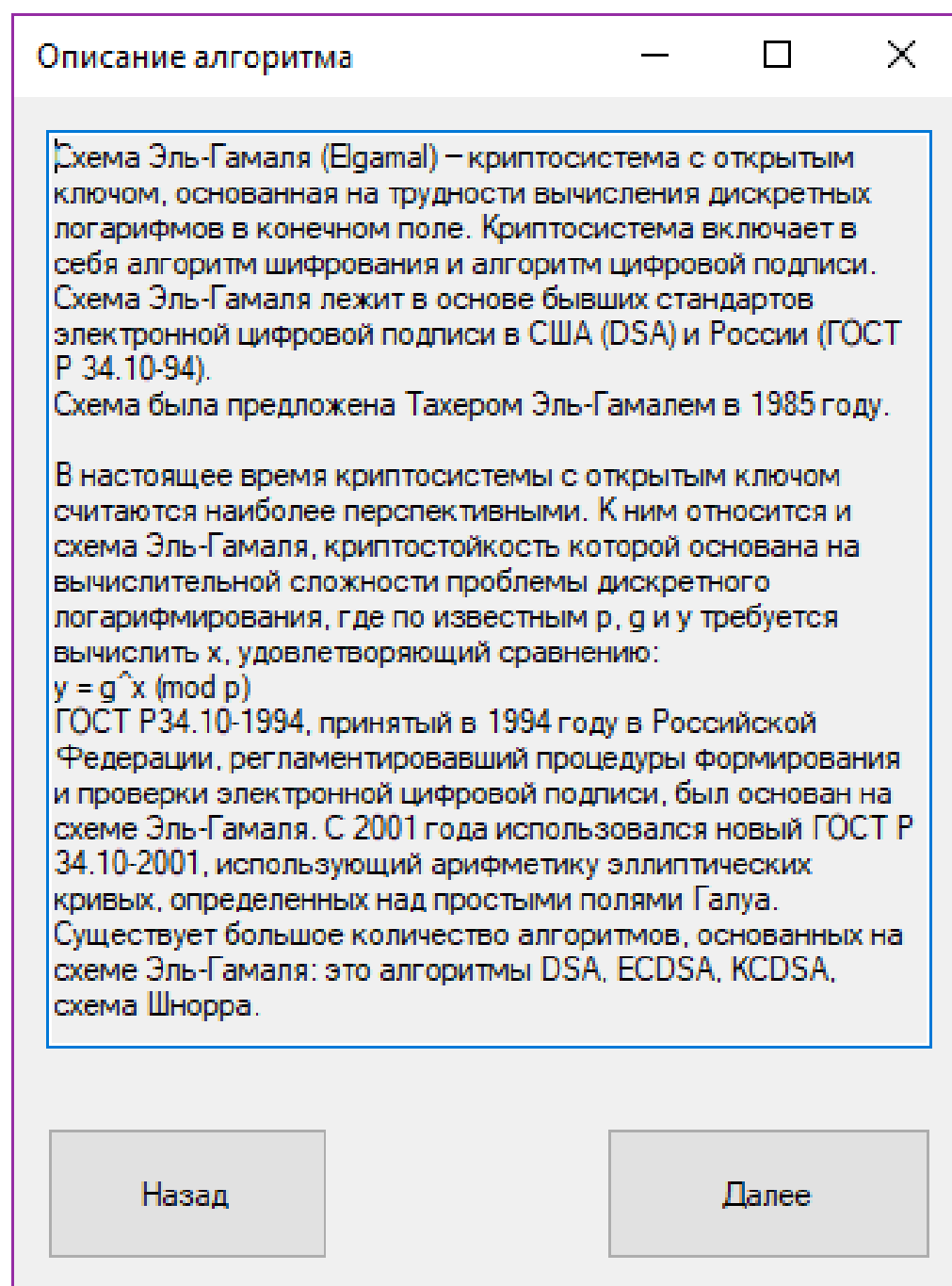


Рис. 25. Общая информация о криптосистеме

На шестом шаге (рис. 26) мы видим конкретный пример генерирования ключей криптосистемы. Числа  $p$  и  $x$  можно как вводить с клавиатуры, так и случайно сгенерировать.

Генерация ключей

—

□

×

Ключи в схеме Эль-Гамала генерируются следующим образом:

1. Генерируется случайное простое число  $p$ .

Генерировать

611722643447422623837281716387

2. Вычисляется число  $g$ , которое является первообразным корнем числа  $p$ .

Вычислить

2

3. Выбирается целое случайное число  $x$ , такое, что  $1 < x < p$ .

Генерировать

364459084728326874992850904736

4. Вычисляется  $y = g^x \bmod p$ .

Вычислить

532331016690079393158023521900

Тройка чисел  $(p, g, y)$  является открытым ключом схемы Эль-Гамала, а число  $x$  - секретным ключом.

Назад

Далее

Рис. 26. Генерация ключей

На следующем шаге (рис. 27) объясняется алгоритм шифрования по схеме Эль-Гамала.



Шифрование

Итак, по открытому каналу получен открытый ключ  $(g, p, y)$ , со значениями:  
 $g = 2$   
 $p = 611722643447422623837281716387$   
 $y = 532331016690079393158023521900$   
 Теперь получившая открытый ключ сторона может зашифровать сообщение  $M$ .

Введите  $M$ :

Шифрование в схеме Эль-Гамала осуществляется в три этапа.

1. Выбираем сессионный ключ: случайное  $k$ , такое, что  $1 < k < p-1$

2. Вычисляем число  $a = g^k \bmod p$ .

3. Вычисляем число  $b = y^k * M \bmod p$ .

Пара чисел  $(a, b)$  является шифротекстом.

Рис. 27. Шифрование

Следующий шаг (рис. 28) показывает, как с помощью открытого и секретного ключей расшифровать сообщение, введённое и зашифрованное на предыдущем шаге.

Расшифрование

Итак, мы получили шифротекст:  
a = 164495149760850741500294664111  
b = 259783102045393445813194326242  
Также у нас есть открытый ключ:  
g = 2  
p = 611722643447422623837281716387  
y = 532331016690079393158023521900  
и секретный ключ  
x = 364459084728326874992850904736  
Сообщение M можно получить по формуле:  $M = b \cdot (a^x)^{-1} \bmod p$

4241088717127167176401

Или, если перевести в текст:

Таким образом, мы провели все этапы шифрования по схеме Эль-Гамала.

Рис. 28. Расшифрование

На следующем шаге (рис. 29) рассказывается о математической задаче дискретного логарифмирования, её связи с криптоанализом схемы Эль-Гамала, а также предлагается вручную решить два задания. Задания на этом шаге генерируются случайным образом.

Дискретное логарифмирование

Для дешифрования (криптоанализа) перехваченных сообщений, зашифрованных по криптосистеме Эль-Гамала, необходимо также перехватить открытый ключ и подобрать секретный ключ  $x$ , такой, что  $g^x \bmod p = y$ . Задача вычисления такого числа называется задачей дискретного логарифмирования. В данном случае нам необходимо найти логарифм по основанию  $g$  от числа  $y$  по модулю  $p$ .

Попробуйте найти логарифм по основанию 2 и модулю 5 от числа 1:

Ответ:

Логарифм по основанию 5 и модулю 7 от числа 4:

Ответ:

Задача дискретного логарифмирования обладает большой вычислительной сложностью и является одной из основных задач, на которых базируется криптография с открытым ключом. На сегодняшний день не существует алгоритмов, позволяющих вычислить дискретный логарифм в конечном поле за полиномиальное время. Существующие алгоритмы решения этой задачи - такие, как алгоритм Шенкса (он же алгоритм больших и малых шагов), решают задачу за экспоненциальное время. Одна из теоретических возможностей эффективного решения задачи вычисления дискретного логарифма связана с квантовыми вычислениями.

Назад
Далее

Рис. 29. Дискретный логарифм

На следующем шаге (рис. 30) рассказывается о существующих экспоненциальных алгоритмах нахождения дискретного логарифма. Для наглядной демонстрации вычислительной сложности дискретного логарифмирования пользователю предлагается реализовать алгоритм полного перебора для дискретного логарифмирования и убедиться в полной непригодности этого метода даже для сравнительно небольших модулей.

Алгоритмы решения задачи дискретного ло...

Примерами экспоненциальных алгоритмов дискретного логарифмирования являются такие методы как алгоритм полного перебора, алгоритм Гельфонда-Шенкса и ро-метод Полларда. Сложность алгоритма полного перебора можно оценить в  $O(p^2)$  операций, что делает его неприемлемым для криптоанализа даже сравнительно небольших ключей. Для наглядной демонстрации вычислительной трудоёмкости перебора, реализуйте на любом языке программирования алгоритм полного перебора для задачи дискретного логарифмирования и найдите  $x$  в следующих задачах:

$79560^x = 182693 \bmod 68831671$   
 Ответ:

$72547^x = 22520254 \bmod 58656431$   
 Ответ:

Рис. 30. Алгоритмы дискретного логарифмирования

На следующих трёх шагах (рис. 31–33) рассказывается об одном из алгоритмов дискретного логарифмирования - алгоритме Гельфонда-Шенкса, также известном как алгоритм больших и малых шагов. На первом из этих шагов пользователь получает общую информацию об алгоритме Гельфонда-Шенкса.

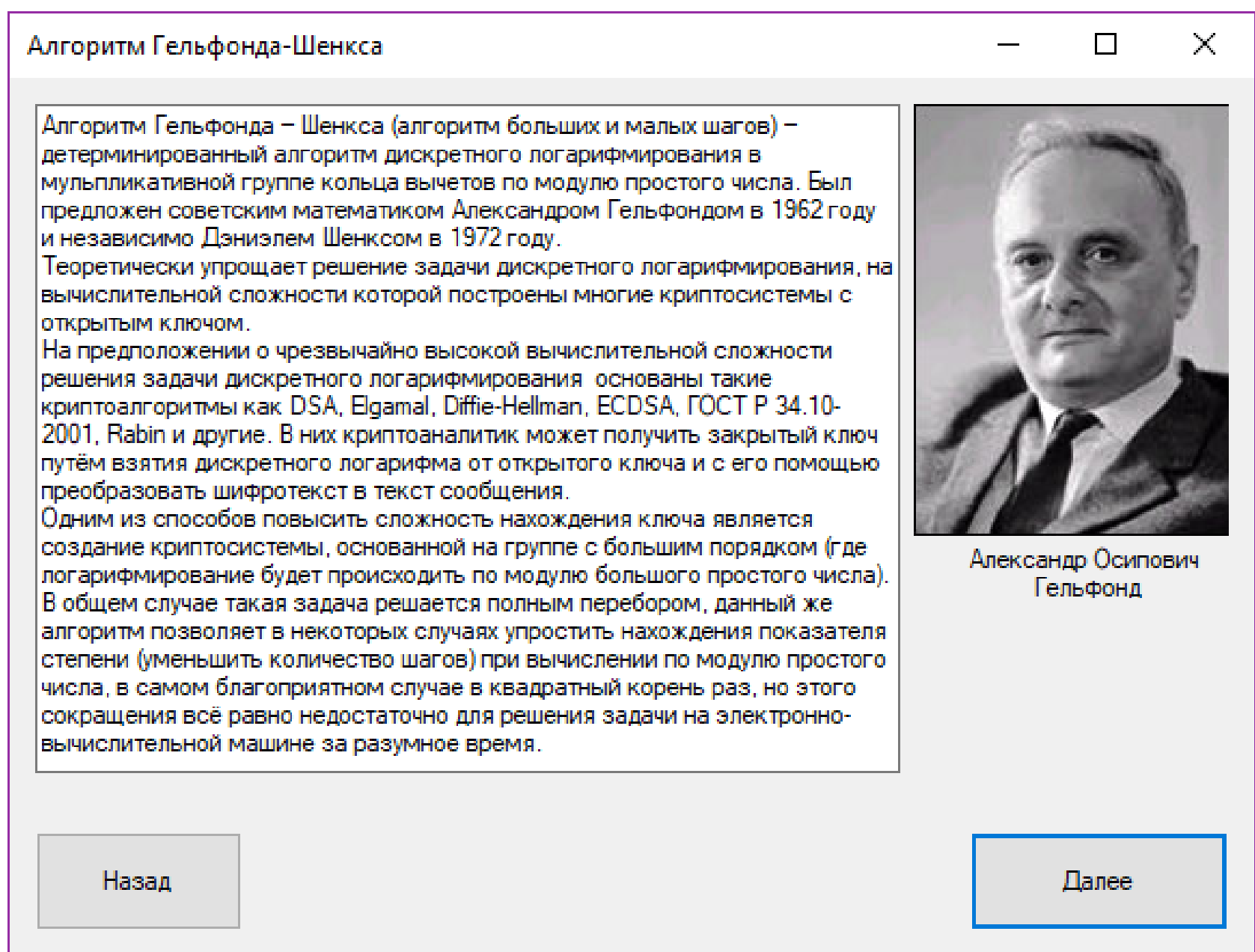


Рис. 31. Общая информация об алгоритме Гельфонда-Шенкса

На следующем шаге (рис. 32) пользователь знакомится с математическим обоснованием алгоритма Гельфонда-Шенкса и, на ещё одном шаге (рис. 33), пользователю демонстрируются шаги алгоритма Гельфонда-Шенкса, записанные превдокодом.

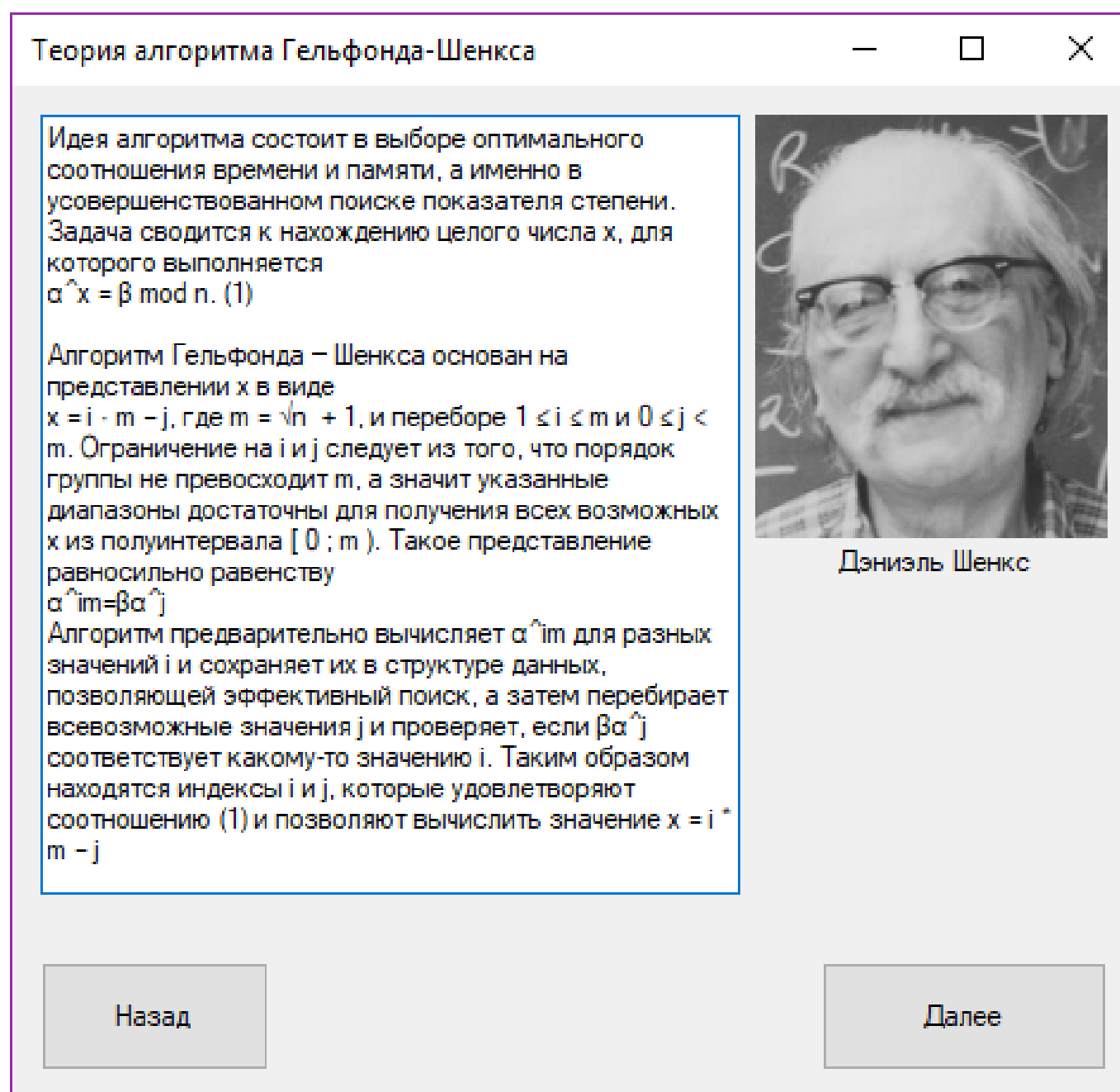


Рис. 32. Математическое обоснование алгоритма Гельфонда-Шенкса

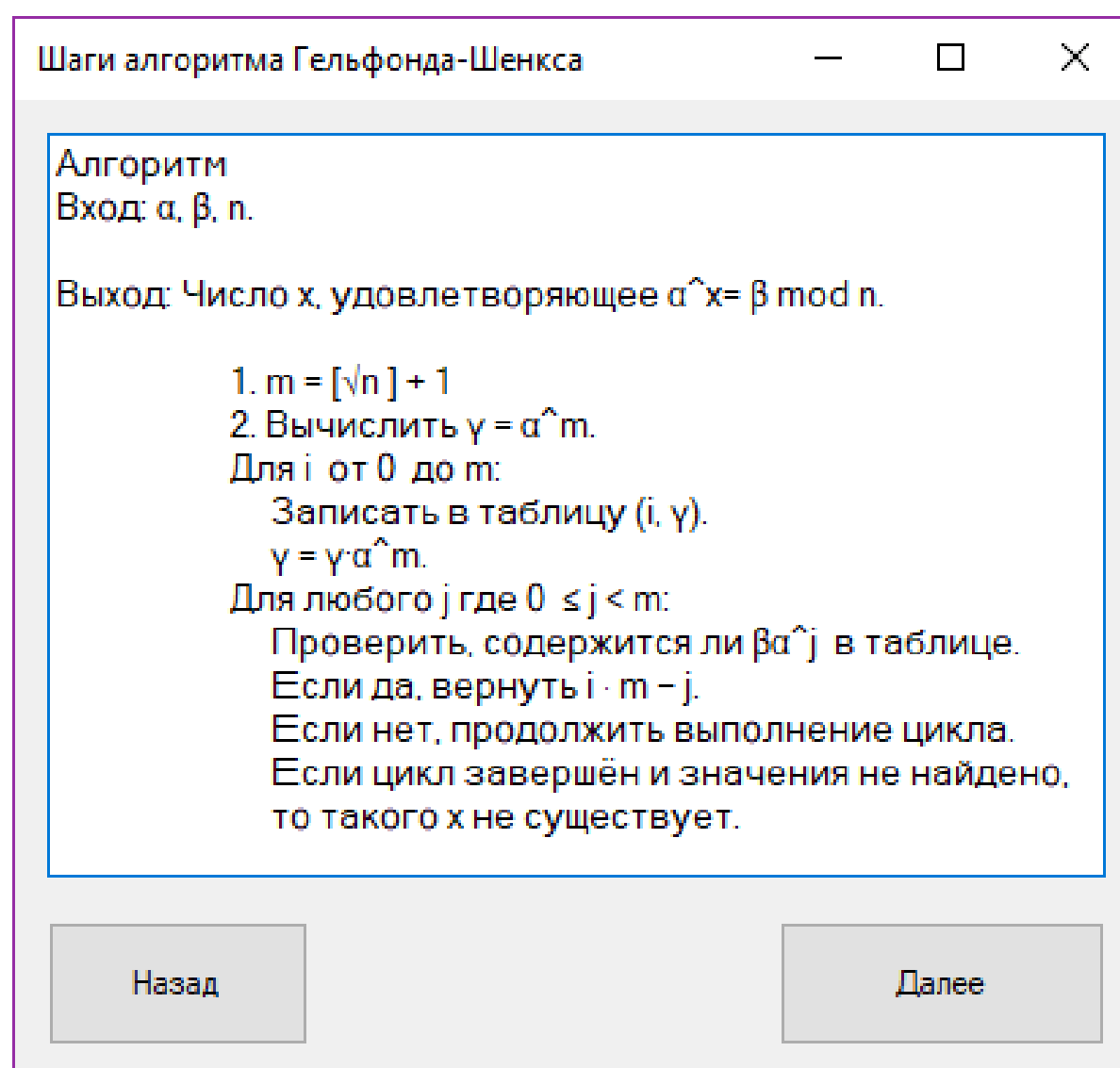


Рис. 33. Псевдокод алгоритма Гельфонда-Шенкса

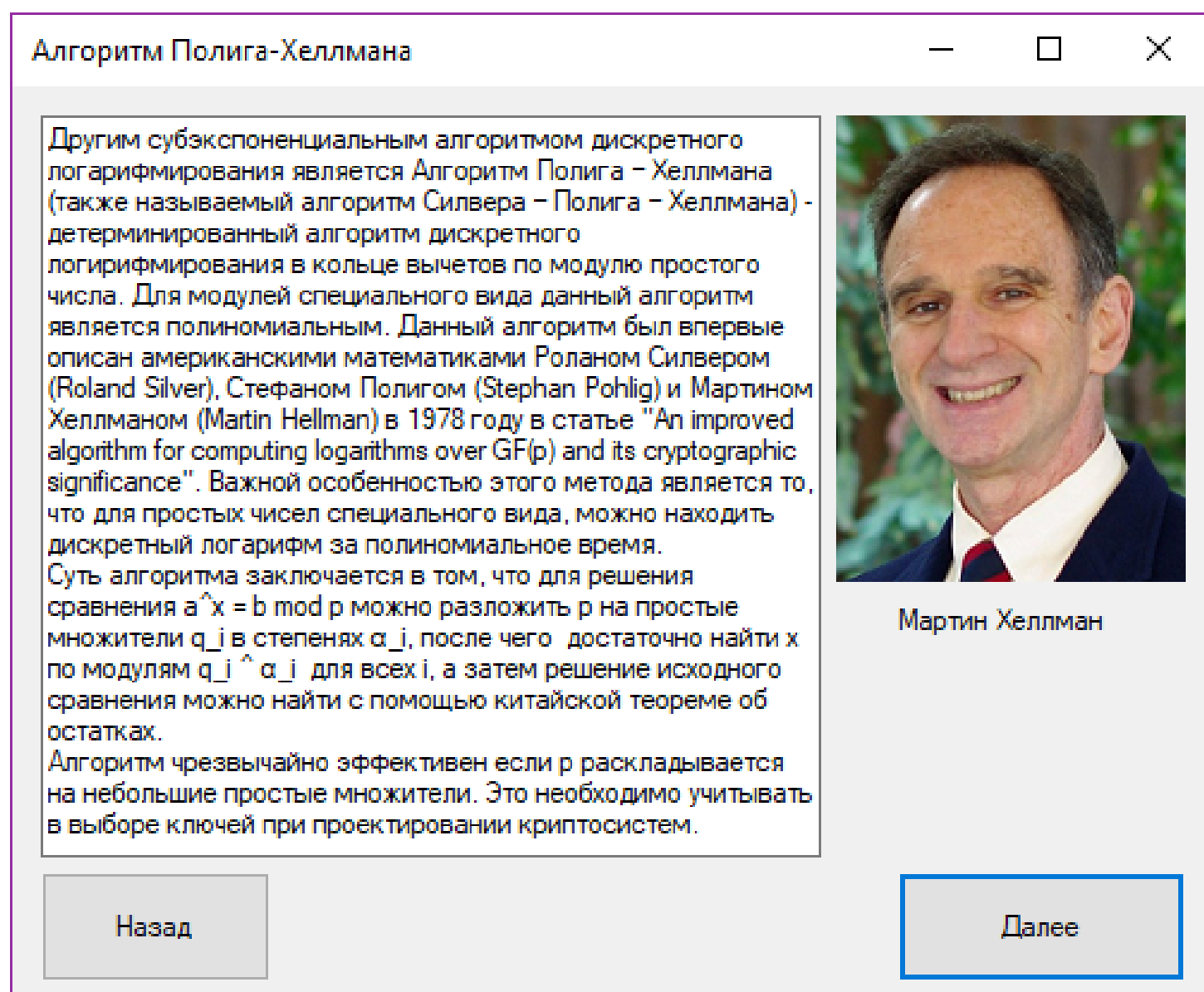


Рис. 34. Общая информация об алгоритме Полига-Хеллмана

На следующих трёх шагах (рис. 34–36) рассказывается о другом алгоритме дискретного логарифмирования - алгоритме Силвера-Полига-Хеллмана. На первом из этих шагов пользователь получает общую информацию об этом алгоритме.

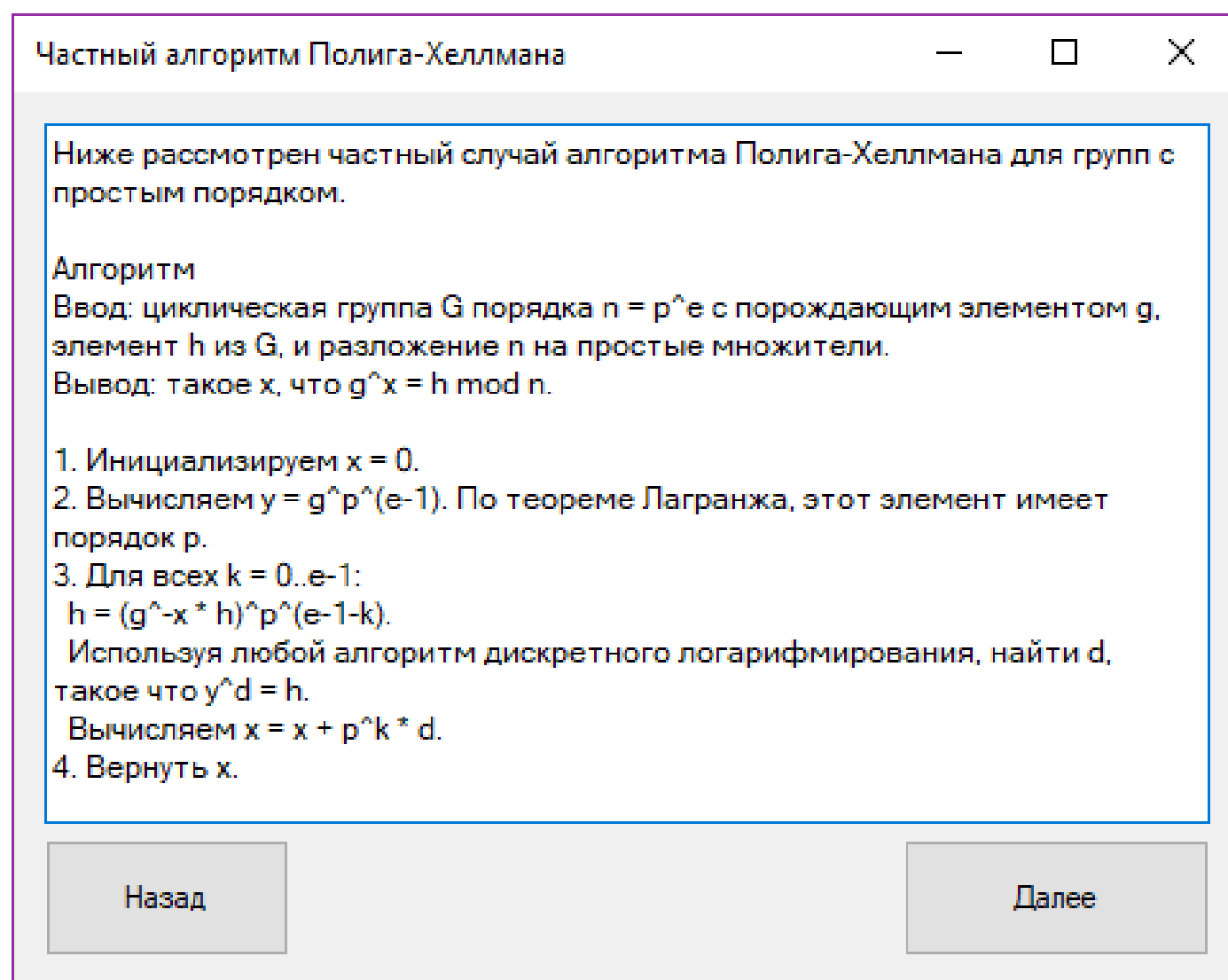


Рис. 35. Частный случай алгоритма Полига-Хеллмана

На втором из этих шагов обучения (рис. 35) демонстрируется псевдокод частного случая алгоритма Полига-Хеллмана, примененного только для групп с простым порядком. Этот алгоритм используется в общем алгоритме Сильвера-Полига-Хеллмана.



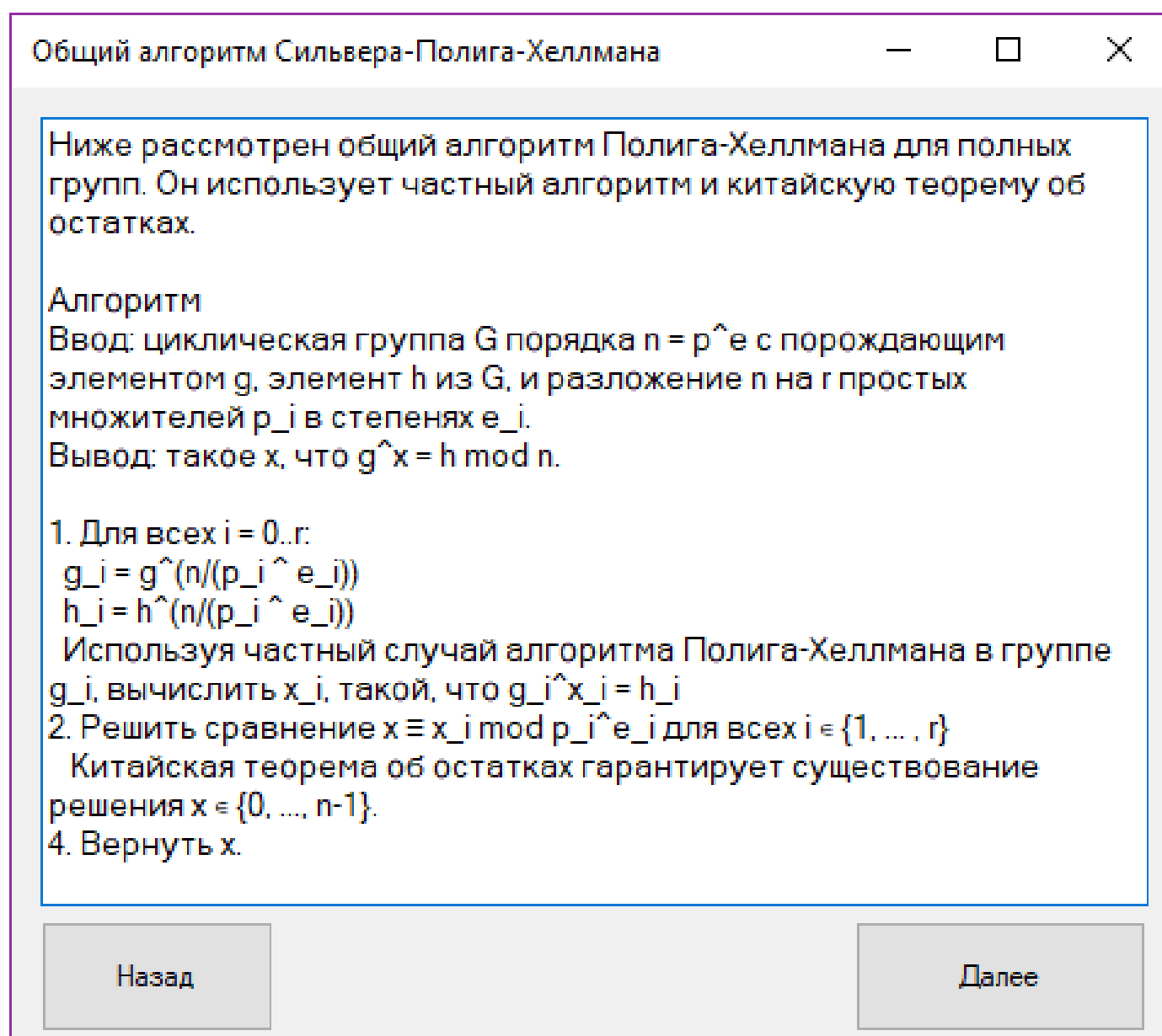


Рис. 36. Общий случай алгоритма Полига-Хеллмана

На третьем из этих шагов (рис. 36) обучения демонстрируется псевдокод общего случая алгоритма Полига-Хеллмана, использующего рассмотренный выше частный алгоритм Сильвера-Полига-Хеллмана и китайскую теорему об остатках.

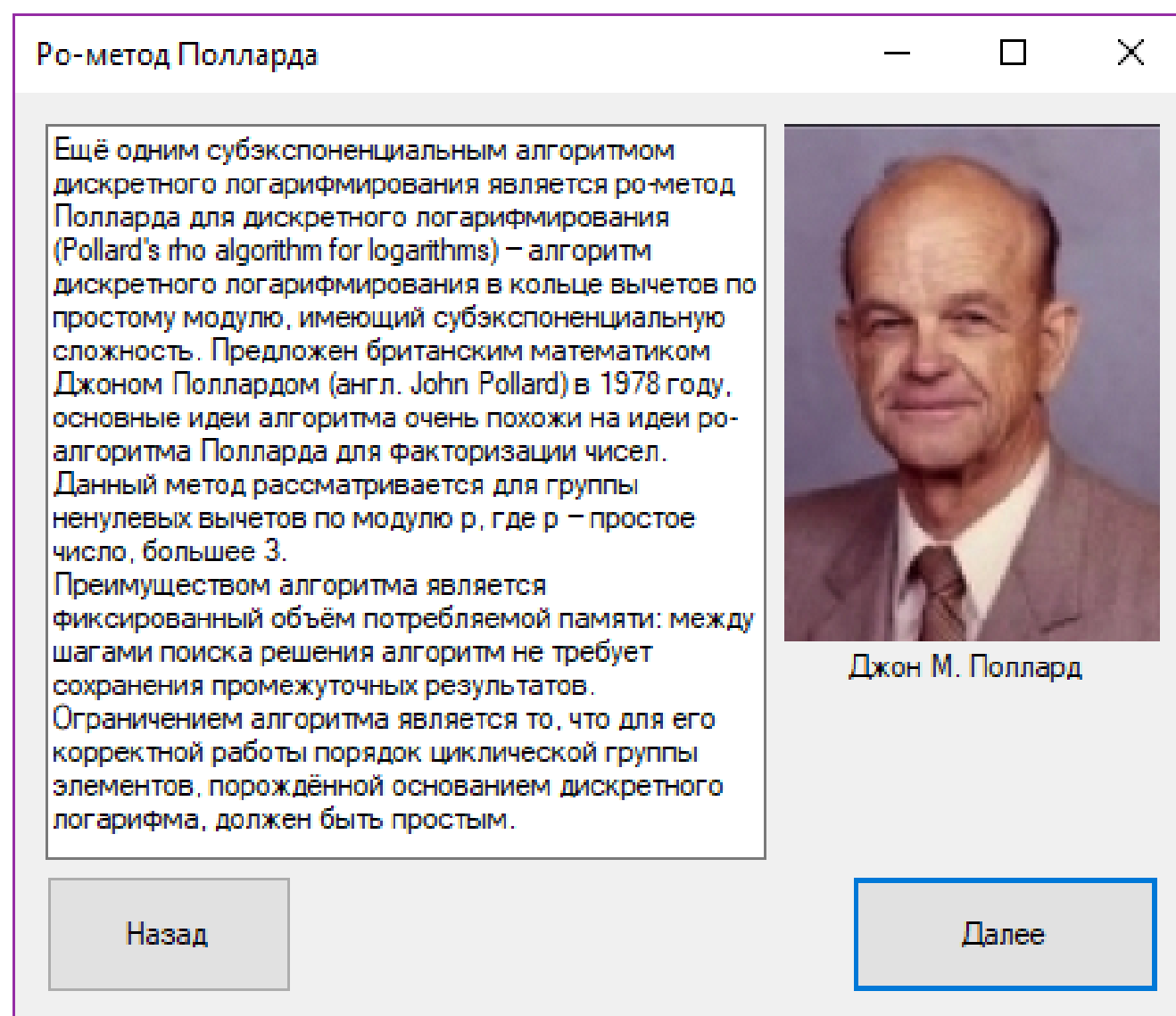


Рис. 37. Общая информация о  $\rho$ -методе Полларда

На следующем шаге обучения (рис. 37) пользователь знакомится с общей информацией об ещё одном алгоритме дискретного логарифмирования -  $\rho$ -методе Полларда, его преимуществах и ограничениях.

На следующих пяти шагах (рис. 38–42) пользователю предлагается пройти небольшое теоретическое по пройденным темам. Тестирование состоит из десяти вопросов с четырьмя вариантами ответов для каждого из них.

Тест, часть 1

Вопрос 1

Функция Эйлера  $\varphi(n)$  - мультипликативная арифметическая функция, равная...

- ☒ Количество целых неотрицательных чисел, меньших  $n$  и взаимно простых с ним
- ☐ Количество целых неотрицательных чисел, меньших или равных  $n$  и взаимно простых с ним
- ☐ Количество целых неотрицательных простых чисел, меньших  $n$
- ☐ Количество действительных чисел, меньших  $n$  и взаимно простых с ним

Вопрос 2

В модулярной арифметике число  $x$  называется величиной, обратной числу  $a$  по модулю  $m$ , если выполнено:

- ☐  $a = x \bmod m$
- ☐  $xm = 1 \bmod a$
- ☒  $ax \bmod m = 1$
- ☐  $am = 1 \bmod x$

Назад

Далее

Рис. 38. Тест, часть 1

Первая часть тестирования (рис. 38) включает в себя вопросы, касающиеся определения функции Эйлера и определения обратного по модулю числа.

Тест, часть 2

Вопрос 3

Если известна функция Эйлера  $\phi(m)$ , то  $a^{-1} \bmod m$  может быть вычислено по формуле:

- ☐  $a * (\phi(m) - 1) \bmod m$
- ☐  $a^{\phi(m)} \bmod m$
- ☒  $a^{(\phi(m) - 1)} \bmod m$
- ☐  $a^{(\phi(m-1))} \bmod m$

Вопрос 4

Что представляет собой шифротекст в криптосистеме Эль-Гамала?

- ☐ Одно целое неотрицательное число
- ☒ Два целых неотрицательных числа
- ☐ Число с количеством десятичных разрядов, равным количеству букв в открытом тексте
- ☐ Число с количеством шестнадцатичных разрядов, равным количеству букв в открытом тексте

Назад Далее

Рис. 39. Тест, часть 2

Вторая часть тестирования (рис. 39) включает в себя два вопроса. Первый вопрос касается вычисления обратного числа по модулю с помощью функции Эйлера, второй вопрос касается вида шифротекста в криптосистеме Эль-Гамала.

Тест, часть 3

— □ ×

Вопрос 5

На какой вычислительной проблеме основана криптостойкость схемы Эль-Гамала?

☐ На сложности факторизации больших чисел

☐ На сложности вычисления первообразных корней в конечном поле

☐ На сложности вычисления дискретного логарифма в группе точек эллиптической кривой

☒ На сложности вычисления дискретного логарифма в конечном поле

Вопрос 6

Для натурального числа  $n$  функция Эйлера может быть вычислена по формуле:

☐  $\varphi\left(\prod_{i=1}^{n-1} p_i^{k_i}\right) = \prod_{i=1}^{n-1} (p_i^{k_i} - p_i^{k_i-1})$

☐  $\varphi\left(\prod_{i=1}^n p_i^{k_i}\right) = \prod_{i=1}^n (p_i^{k_i} - p_i^{k_i+1})$

☐  $\varphi\left(\prod_{i=1}^n p_i^{k_i}\right) = \prod_{i=1}^n (p_i^{k_i} + p_i^{k_i-1})$

☒  $\varphi\left(\prod_{i=1}^n p_i^{k_i}\right) = \prod_{i=1}^n (p_i^{k_i} - p_i^{k_i-1})$

Назад

Далее

Рис. 40. Тест, часть 3

Следующие два вопроса (рис. 40) касаются оснований криптографической стойкости криптосистемы Эль-Гамала и общей формулы вычисления функции Эйлера.

Тест, часть 4

Вопрос 7

Какой вычислительной сложностью обладает алгоритм Гельфонда-Шенкса?

☐ Субэкспоненциальной

☐ Линейной

☐ Полиномиальной

☒ Экспоненциальной

Вопрос 8

Стойкость какого из следующих криптографических алгоритмов не основана на вычислительной сложности дискретного логарифма?

☐ ECDSA

☒ RSA

☐ Diffie-Hellman

☐ ГОСТ Р 34.10-2001

Назад

Далее

Рис. 41. Тест, часть 4

Седьмой вопрос посвящён вычислительной сложности алгоритма Гельфонда-Шенкса. Восьмой вопрос посвящён другим криптосистемам, основанным на криптографической стойкости дискретного логарифмирования (рис. 41).

Тест, часть 5

Вопрос 9

Какой из приведённых алгоритмов дискретного логарифмирования применим только для групп с порядком специального вида?

☐ Алгоритм Сильвера-Полига-Хеллмана

☒ Р-метод Полларда

☐ Алгоритм Гельфонда-Шенкса

☐ Ни один

Вопрос 10

Какой из следующих алгоритмов дискретного логарифмирования обладает полиномиальной сложностью в общем случае?

☐ Алгоритм Сильвера-Полига-Хеллмана

☐ Р-метод Полларда

☐ Алгоритм Гельфонда-Шенкса

☒ Ни один

Назад

Далее

Рис. 42. Тест, часть 5

Наконец, пятая часть теста (рис. 42) посвящена границам применимости и вычислительной сложности алгоритмов дискретного логарифмирования.

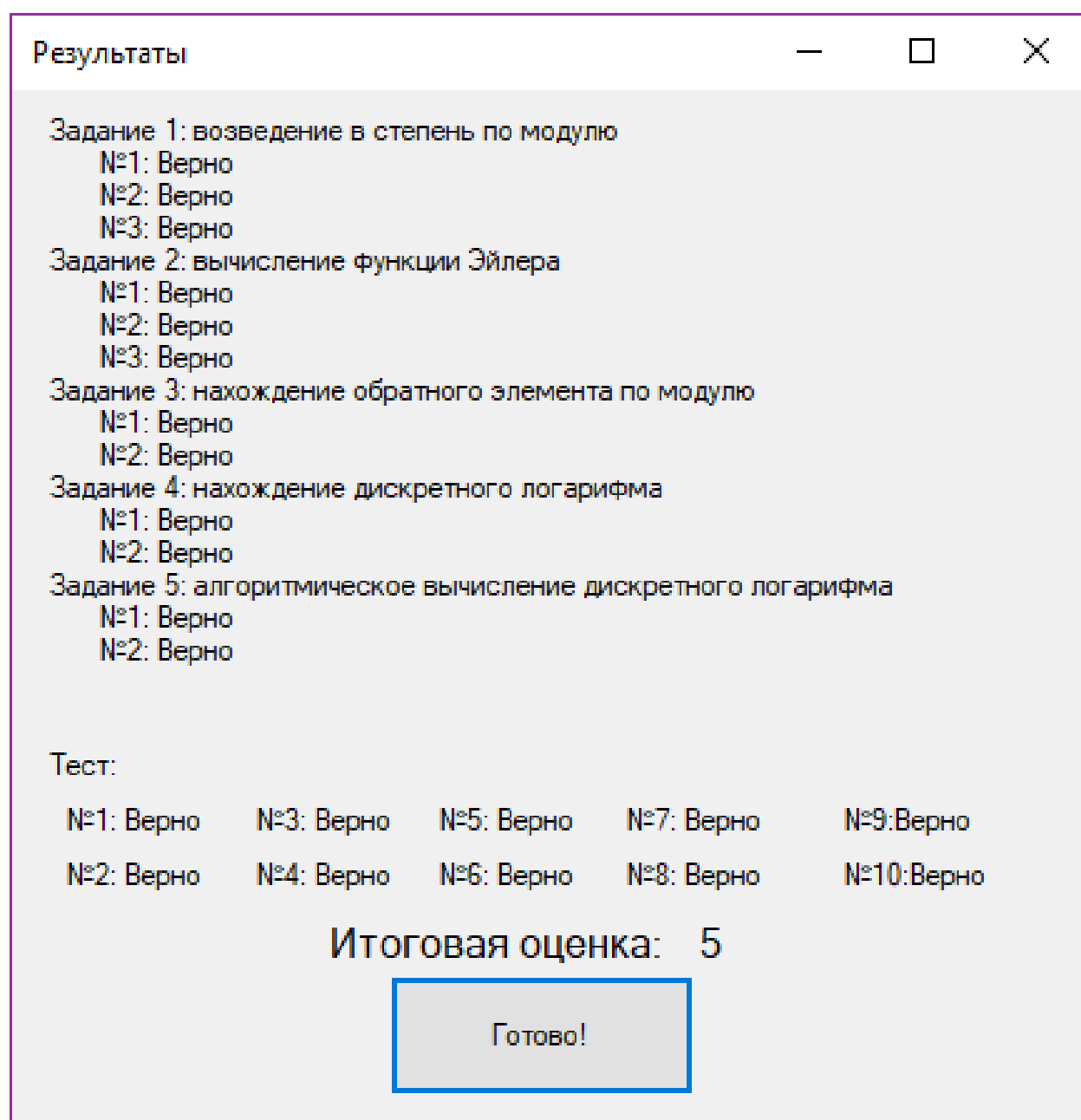


Рис. 43. Результаты проверки знаний

На завершающем шаге обучения (рис. 43) показаны результаты выполненных заданий и ответов на вопросы теста, и пользователь получает оценку своих знаний.

## 5.6. Вызов и загрузка

Вызов программы осуществляется стандартными средствами системы Windows с установленной платформой Microsoft .NET Framework 4.5. Имя загрузочного модуля – ElgamalTutor.exe.

## 5.7. Входные и выходные данные

Входные данные – ответы на поставленные вопросы, выходные данные – результаты тестирования и примеры работы криптосистемы.



## 6. ОПИСАНИЕ СЦЕНАРИЯ ЛАБОРАТОРНОЙ РАБОТЫ

### 6.1. Постановка задачи

1. Ознакомиться с обучающей компьютерной программой El-Gamal\_Tutor.
2. Изучить и привести описание алгоритма Эль Гамала (в соответствии с обозначениями из [1]) с доказательством корректности алгоритма, его достоинствами и недостатками.
3. Зафиксировать (для отчета) последовательность этапов обучения в программе El-Gamal\_Tutor.
4. Провести тестирование программы El-Gamal\_Tutor с целью выявления ошибок и недочетов.
5. С помощью математического пакета прикладных программ произвести шифрование и расшифрование сообщения, заданного в виде одного блока открытого текста. При этом длина ключей должна удовлетворять условиям:  $|p|, |\delta| \geq 80$ ,  $|r|, |\alpha| \geq 40$ . Ключи описываются соотношениями:  $K = (p, \alpha, \beta, \delta) : \alpha^\delta \equiv \beta \pmod{p}$ , где  $K = (k_O; k_S)$ ;  $k_O = (p, \alpha, \beta)$  – открытый ключ;  $k_S = (\delta)$  – закрытый ключ.
6. Сформулировать и обосновать принципы работы алгоритма Эль Гамала.
7. Одним из методов решения задачи дискретного логарифмирования осуществить криптоанализ заданного шифрованного текста на основе известных составляющих открытого ключа  $(p, \alpha, \beta)$ .
8. Ответить на контрольные вопросы.
9. Составить и защитить отчет о проделанной работе.

### 6.2. Содержание отчета о выполнении лабораторной работы

1. Постановка задачи
2. Описание криптосистемы Эль Гамала.

3. Последовательность этапов и результаты обучения с использованием программы El-Gamal\_Tutor.
4. Выявление ошибок и недочетов в обучающей программе El-Gamal\_Tutor.
5. Результаты шифрования и расшифрования с использованием ППП Maple.
6. Принципы работы алгоритма Эль Гамала.
7. Последовательность этапов и результаты криптоанализа.
8. Ответы на контрольные вопросы.
9. Выводы
10. Библиография

## ЗАКЛЮЧЕНИЕ

Практически в каждой коммерческой, военной и государственной отрасли требуется наличие системы, позволяющей шифровать данные, делая информацию, заложенную в эти данные, невозможной для воспроизведения обычным пользователем. Для получения нужных для этого кадров необходим эффективный процесс обучения специалистов в данной отрасли.

В обучении будущих специалистов в любой области немаловажную роль играет использование наглядных средств обучения, эксперимент (в том числе вычислительный) и проверка полученных знаний на практике. В современной образовательной системе успешно используются обучающие программы, лучшие из которых, как правило, сочетают в себе все эти три фактора. В настоящей работе был предложен вариант лабораторной работы по защите информации, специально для которой была разработана обучающая программа, проектировавшаяся в первую очередь исходя из всех вышеуказанных принципов.

Программа была представлена на нескольких научно-практических конференциях [11–14], лабораторная работа была представлена на двух научно-практических конференциях [13, 14]. В настоящее время осуществляется государственная регистрация программы.

## СПИСОК ЛИТЕРАТУРЫ

1. Воронков Б. Н. Криптографические методы защиты информации: учебное пособие / Б. Н. Воронков. – Воронеж: Издательско-полиграфический центр Воронежского государственного университета, 2008. – 59 с.
2. Схема Эль-Гамала / Википедия [текст]. – (URL: [https://ru.wikipedia.org/wiki/Схема\\_Эль-Гамала](https://ru.wikipedia.org/wiki/Схема_Эль-Гамала)) (дата обращения 13.05.2018).
3. Дискретное логарифмирование / Википедия [текст]. – (URL: [https://ru.wikipedia.org/wiki/Дискретное\\_логарифмирование](https://ru.wikipedia.org/wiki/Дискретное_логарифмирование)) (дата обращения 06.05.2018).
4. ГОСТ 27463-87. Системы обработки информации. 7-битные кодированные наборы символов (с Изменением N 1) / Электронный фонд правовой и нормативно-технической документации [текст]. – (URL: <http://docs.cntd.ru/document/gost-27463-87>) (дата обращения 17.03.2018)
5. Вычисления, полезные в криптографии : Crypto03.exe, v. 10.11.03 [электронный ресурс]. – Ставрополь : КубГТУ, 2003.
6. Витер В. Обучающая программа El-Gamal. Версия 1.0 / В. Витер [электронный ресурс], 1999 г. – (vvvslava@hotmail.ru).
7. Воронков Б. Н. Обучающая компьютерная программа для изучения российского стандарта криптографического преобразования / Б. Н. Воронков, И. И. Проскурин // Современные информационные технологии и ИТ-образование. Сборник избранных трудов 6-ой международной НПК (г. Москва, 12 – 14 декабря 2011 г.). – М.: ИНТУИТ.РУ, 2011. – С. 121 – 127.
8. Кабанов Е. В. Программа обучения алгоритму шифрования DES / Е. В. Кабанов, М. В. Прокопов [электронный ресурс], 2001г. – (URL: <http://www.blackw.des.ru>, nexus@mail.ru) (дата обращения 28.09.2009).
9. Buchmann J. On some computational problems in finite abelian groups / Buchmann J., Jacobson M. J., Teske E. Mathematics of Computation, 1997, 220(66), pp. 1663-1687.

10. Алгоритм Адлемана / Википедия [текст]. – (URL: [https://ru.wikipedia.org/wiki/Алгоритм\\_Адлемана](https://ru.wikipedia.org/wiki/Алгоритм_Адлемана)) (дата обращения 08.03.2018).
11. Ковун В. А. Электронная обучающая программа El-Gamal\_Tutor / В. А. Ковун, Б. Н. Воронков // Информатика: проблемы, методология, технологии: сборник материалов XVI международной научно-методической конференции, г. Воронеж, 11 – 12 февраля 2016 г. – Воронеж: Издательство «Научно-исследовательские публикации», 2016. – 3078 с. – Раздел 6. VII международная школа-конференция «Информатика в образовании». – С. 171 – 176.
12. Ковун В. А. Криптосистема Эль Гамала. Лабораторная работа / В. А. Ковун, Б. Н. Воронков // Информатика: проблемы, методология, технологии: сборник материалов XVII международной научно-методической конференции, г. Воронеж, 9 – 10 февраля 2017 г.: в 5-ти томах. – Воронеж: Издательство «Научно-исследовательские публикации», 2017. – Т. 5. Информатика в образовании: материалы VII Школы-конференции. Секция 6: «Применение информационных технологий в преподавании различных дисциплин». – С. 169 – 174.
13. Ковун В. А. Криптоанализ в обучающей программе El-Gamal\_Tutor / В. А. Ковун, Б. Н. Воронков // Информатика: проблемы, методология, технологии: сборник материалов XVIII международной научно-методической конференции, г. Воронеж, 8 – 9 февраля 2018 г.: в 7-ти томах. – Воронеж: Издательство «Научно-исследовательские публикации» (ООО «Вэлборн»), 2018. – Т. 7. – С. 194 – 198.
14. Ковун В. А. Алгоритм Эль Гамала. Лабораторная работа / В. А. Ковун, Б. Н. Воронков // Математика, информационные технологии, приложения: сборник трудов Межвузовской научной конференции молодых ученых и студентов, 23 апреля 2018 г. – Воронеж : Издательство «Научно-исследовательские публикации», 2018 – С. 46 – 60.

## ПРИЛОЖЕНИЕ

Текст программы “El-Gamal\_Tutor”.

Электронная обучающая программа для изучения асимметричной крип-  
тосистемы Эль-Гамала.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class AboutForm : Form
    {
        public AboutForm()
        {
            InitializeComponent();
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;

namespace ElgamalTutor
{
    static class Answers
    {
        public static int formToShow = 0;
        public static bool tutorialEnded = false;
        public static bool interruptTutorial = false;
    }
}
```

```

public static BigInteger p = new BigInteger(0);
public static BigInteger g = new BigInteger(0);
public static BigInteger x = new BigInteger(0);
public static BigInteger y = new BigInteger(0);
//public static BigInteger k = new BigInteger(0);
public static BigInteger a = new BigInteger(0);
public static BigInteger b = new BigInteger(0);

public static bool[] modpowAnswers = new bool[3];
public static bool[] EulerAnswers = new bool[3];
public static bool[] ReverseAnswers = new bool[2];
public static bool[] DiscreteLogAnswers = new bool[2];
public static bool[] AlgorithmicDLOGAnswers = new bool[2];
public static bool[] TestAnswers = new bool[10];
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Numerics;
using System.Windows.Forms;

namespace ElgamalTutor
{
    static class BabyStepGiantStep
    {
        public static BigInteger sqrt(BigInteger n)
        {
            if (n == 0) return 0;

            BigInteger sqroot = 0;

            if (n > 0)
            {
                int lengthInBits = Convert.ToInt32(
                    Math.Ceiling(BigInteger.Log(n, 2))
                );
            }
        }
    }
}

```

```

    root = BigInteger.One << (lengthInBits / 2);

    while (!isSqrtOf(n, sqroot))
    {
        root += n / sqroot;
        root /= 2;
    }

    return sqroot;
}

throw new ArithmeticException("NaN");
}

private static Boolean isSqrtOf(BigInteger n, BigInteger sqroot
    )
{
    BigInteger lowerBound = sqroot * sqroot;
    BigInteger upperBound = (sqroot + 1) * (sqroot + 1);

    return (n >= lowerBound && n < upperBound);
}

public static BigInteger bsgs(BigInteger a, BigInteger b,
    BigInteger m)
{
    if (a == b)
        return 1;

    var map = new Dictionary<BigInteger, BigInteger>();
    BigInteger ans, n, i;
    ans = 1;

    n = sqrt(m) + 1; // 1

    for (i = n; i >= 1; --i) // 3
    {
        BigInteger curKey = BigInteger.ModPow(a, i * n, m);
        if (!(map.ContainsKey(curKey)))

```



```

        map.Add(curKey, i);
    else
        map[curKey] = i;
    }

    for (i = 0; i <= n; ++i) // 4
    {
        BigInteger cur = (BigInteger.ModPow(a, i, m) * b) % m;
        if (map.ContainsKey(cur))
        {
            ans = map[cur] * n - i;
            if (ans < m) return ans;
        }
    }

    return -1;
}

public static BigInteger bsgs2(BigInteger a, BigInteger b,
    BigInteger m, ProgressBar progress = null)
{
    if (a == b)
        return 1;

    var map = new Dictionary<BigInteger, BigInteger>();
    BigInteger ans, n, i, an, curKey;
    ans = 1;
    an = 1;

    n = sqrt(m) + 1; // 1
    an = BigInteger.ModPow(a, n, m);
    //for (i = 0; i < n; ++i)
    //    an = (an * a) % m;

    BigInteger tick = n / 50; // for progressbar
    if (tick == 0)
        tick = 1;
    // int progressQuantum = (progress.Width / 100);

    for (i = 1, curKey = an; i <= n; ++i) // 3

```

```

    {
        if (!(map.ContainsKey(curKey)))
            map.Add(curKey, i);
        curKey = (curKey * an) % m;

        if (progress != null) // progressbar
            if (i % tick == 0) // progressbar!
                progress.Increment(progress.Width / 100); //
    }

    for (i = 0, curKey = b; i <= n; ++i) // 4
    {
        if (map.ContainsKey(curKey))
        {
            ans = map[curKey] * n - i;
            if (ans < m) return ans;
        }
        curKey = (curKey * a) % m;

        if (progress != null) // progressbar
            if (i % tick == 0) // progressbar!
                progress.Increment(progress.Width / 100); //

    }

    return -1;
}

}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Numerics;

namespace ElgamalTutor
{
    static class BigIntegerSqrt

```

```

{

private static BigInteger modulo = BigInteger.Pow(BigInteger.
    Parse("10000000000000000000000000000000"), 10000);

public static BigInteger Sqrt(this BigInteger n)
{
    if (n == 0) return 0;
    if (n > 0)
    {
        int bitLength = Convert.ToInt32(Math.Ceiling(BigInteger.Log
            (n, 2)));
        BigInteger root = BigInteger.One << (bitLength / 2);

        while (!isSqrt(n, root))
        {
            root += n / root;
            root /= 2;
        }

        return root;
    }

    throw new ArithmeticException("NaN");
}

private static Boolean isSqrt(BigInteger n, BigInteger root)
{
    BigInteger lowerBound = root * root;
    BigInteger upperBound = (root + 1) * (root + 1);

    return (n >= lowerBound && n < upperBound);
}

public static BigInteger Power(this BigInteger n, BigInteger
    exponent)
{
    // return BigInteger.Pow(n, (int)exponent);
    return BigInteger.ModPow(n, exponent, modulo);
}
}

```

```

}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class CalcfiForm : Form
    {
        BigInteger input = new BigInteger(0);
        public CalcfiForm()
        {
            InitializeComponent();

            private void calcBtn_Click(object sender, EventArgs e)
            {
                this.Enabled = false;
                input = BigInteger.Parse(inputBox.Text);
                input = CryptoMath.phi(input);
                outputBox.Text = input.ToString();
                label2.Visible = true;
                outputBox.Visible = true;
                this.Enabled = true;
            }

            private void inputBox_TextChanged(object sender, EventArgs e)
            {
                label2.Visible = false;
                outputBox.Visible = false;
            }
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    class CryptoMath
    {
        public static Random rand = new Random();
        public static int MILLER_RABIN_ROUNDS = 4000;
        public const long DIGITS = 20;

        public static BigInteger phi(BigInteger n)
        {
            BigInteger result = n;
            BigInteger i = new BigInteger(2);
            for (i = 2; i * i <= n; ++i)
                if (n % i == 0)
                {
                    while (n % i == 0)
                        n /= i;
                    result -= result / i;
                }
            if (n > 1)
                result -= result / n;
            return result;
        }

        //public static BigInteger powmod(BigInteger a, BigInteger b,
        //    BigInteger m)
        //{
        //    BigInteger r=1;
        //    a%=m;

```

```
// while (b>0)
// {
//     if (b%2 ==1) r=(r*a)%m;
//     a=(a*a)%m;
//     b>>=1;
// }
// return r;
//}
```

```
public static bool isPrime(BigInteger n)
{
    if (n == 2 || n == 3 || n == 5 || n == 7)
        return true;
    if ((n % 60) % 5 == 0)
        return false;
    if ((n % 60) % 3 == 0)
        return false;
    if ((n % 60) % 2 == 0)
        return false;
    //тест Миллера-Рабина
    //сначала представим m-1 в виде m-1 = 2^s * t
    BigInteger m = n-1;
    BigInteger s = 0;
    BigInteger t = 1;
    while(m/2 > 0 && m%2 == 0)
    {
        m /= 2;
        s++;
    }
    t = m;
    m = n;
    for (BigInteger i = 0; i < MILLER_RABIN_ROUNDS; i++)
    {
        BigInteger a = rand.Next()%(m-3) + 3;
        BigInteger x = BigInteger.ModPow(a,t,m);
        if (x == 1 || x == m-1) continue;
        for (BigInteger j = 0; j < s-1; j++)
        {
            x = BigInteger.ModPow(x, 2, m);
```

```

        if (x == m-1) break;
        if (x == 1) return false;

    }
    if (x==m-1) continue;
    return false;
}
return true;
}

public static BigInteger calculatePrimitiveRoot(BigInteger p)
{
    List<BigInteger> fact = new List<BigInteger>();
    BigInteger phi = p-1, n = phi;
    for (BigInteger i=2; i*i<=n; ++i)
        if (n % i == 0) {
            fact.Add(i);
            while (n % i == 0)
                n /= i;
        }
    if (n > 1)
        fact.Add(n);
    for (BigInteger res=2; res<=p; ++res)
    {
        bool ok = true;
        for (int i=0; i<fact.Count && ok; ++i)
            ok &= BigInteger.ModPow(res, phi / fact
                [i], p) != 1;
        if (ok) return res;
    }
    return -1;
}

public static BigInteger calculateFakePrimitiveRoot(BigInteger
    p)
{
    List<BigInteger> fact = new List<BigInteger>();
    BigInteger phi = p - 1, n = phi;
    for (BigInteger i = 2; i * i * i * i <= n; ++i)
        if (n % i == 0)
        {

```

```

        fact.Add(i);
        while (n % i == 0)
            n /= i;
    }
    if (n > 1)
        fact.Add(n);
    for (BigInteger res = 2; res <= p; ++res)
    {
        bool ok = true;
        for (int i = 0; i < fact.Count && ok; ++i)
            ok &= BigInteger.ModPow(res, phi / fact[i], p) != 1;
        if (ok) return res;
    }
    return -1;
}

public static BigInteger genSimpleRand(long decDigits/*=DIGITS
    */)
{
    BigInteger a = new BigInteger(0);
    BigInteger pow = 1;
    while (!isPrime(a))
    {
        a = 0;
        pow = 1;
        for (long i = 0; i < decDigits; i++)
        {
            BigInteger rnd = (rand.Next(8)+1) * pow;
            a += rnd;
            pow *= 10;
        }
    }
    return a;
}

public static BigInteger genRand(long decDigits/* = DIGITS*/)
{
    BigInteger a = new BigInteger(0);
    BigInteger pow = 1;
    a = 0;
    pow = 1;

```



```

        for (long i = 0; i < decDigits; i++)
        {
            BigInteger rnd = rand.Next(10) * pow;
            a += rnd;
            pow *= 10;
        }
        return a;
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class DecryptForm : Form
    {
        private BigInteger p = new BigInteger(0);
        private BigInteger g = new BigInteger(0);
        private BigInteger x = new BigInteger(0);
        private BigInteger y = new BigInteger(0);
        private BigInteger a = new BigInteger(0);
        private BigInteger b = new BigInteger(0);
        private BigInteger M = new BigInteger(0);
        private long digits = 10;
        public DecryptForm()
        {
            InitializeComponent();
            digitsBox.Text = digits.ToString();
        }

        private void genpgBtn_Click(object sender, EventArgs e)

```

```

{
    string pstr;
    switch (digitsBox.Text)
    {
        case "p40":
            pstr = "7524548124131735373612526345481757634861";
            BigInteger.TryParse(pstr, out p);
            g = 7;
            break;
        case "p41":
            pstr = "61571157514825784611687343327637886854113";
            BigInteger.TryParse(pstr, out p);
            g = 3;
            break;
        default:
            p = CryptoMath.genSimpleRand(digits);
            g = CryptoMath.calculateFakePrimitiveRoot(p);
            break;
    }
    pBox.Text = p.ToString();
    gBox.Text = g.ToString();
}

private void genxBtn_Click(object sender, EventArgs e)
{
    if (p > 2)
        do
        {
            x = CryptoMath.genRand(digits);
            while (x == 0 || x >= p);
        }
        while (xBox.Text == x.ToString());
}

private void genyBtn_Click(object sender, EventArgs e)
{
    if (p > 2)
        y = BigInteger.ModPow(g, x, p);
    yBox.Text = y.ToString();
}

private void decryptBtn_Click(object sender, EventArgs e)
{

```

```

    Encoding encoding = Encoding.GetEncoding(1251);
    BigInteger apx = BigInteger.ModPow(a, x, p);
    apx = BigInteger.ModPow(apx, p - 2, p);
    M = (b * apx) % p;
    byte[] textBytes = M.ToByteArray();
    MBox.Text = encoding.GetString(textBytes);
}

private void digitsBox_TextChanged(object sender, EventArgs e)
{
    int i;
    if (int.TryParse(digitsBox.Text, out i) && i >= 0)
        digits = i;
}

private void pBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(pBox.Text, out p);
}

private void gBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(gBox.Text, out g);
}

private void xBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(xBox.Text, out x);
}

private void yBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(yBox.Text, out y);
}

private void aBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(aBox.Text, out a);
}

private void bBox_TextChanged(object sender, EventArgs e)

```

```

        {
            BigInteger.TryParse(bBox.Text, out b);
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class DiscreteLogAlgorithmsInfoForm : Form
    {
        static Random r = new Random();
        static BigInteger base1, base2, num1, num2, mod1, mod2, ans1,
            ans2;

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }

        public DiscreteLogAlgorithmsInfoForm()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
            // textBox2.Select(0, 0);
            base1 = new BigInteger(r.Next(100000) + 2);
            base2 = new BigInteger(r.Next(100000) + 2);
            mod1 = CryptoMath.genSimpleRand(8);
            if (mod1 == base1) mod1++;
        }
    }
}

```

```

mod2 = CryptoMath.genSimpleRand(8);
if (mod2 == base2) mod1++;
BigInteger rnd1 = CryptoMath.genRand(100000);
BigInteger rnd2 = CryptoMath.genRand(100000);
num1 = BigInteger.ModPow(base1, rnd1, mod1);
num2 = BigInteger.ModPow(base2, rnd2, mod2);
Task1Label.Text =
    base1.ToString() + "^x  = " + num1.ToString() + " mod " +
        mod1.ToString();
Task2Label.Text =
    base2.ToString() + "^x  = " + num2.ToString() + " mod " +
        mod2.ToString();
}

private void DiscreteLogAlgorithmsInfoForm_Load(object sender,
    EventArgs e)
{

}

private void nextBtn_Click(object sender, EventArgs e)
{
    if (BigInteger.TryParse(AnswerBox1.Text, out ans1) &&
        BigInteger.TryParse(AnswerBox2.Text, out ans2))
    {
        Answers.AlgorithmicDLOGAnswers[0] = (num1 == BigInteger.
            ModPow(base1, ans1, mod1));
        Answers.AlgorithmicDLOGAnswers[1] = (num2 == BigInteger.
            ModPow(base2, ans2, mod2));
        //var newForm = new TutorForm3();
        //newForm.Show();
        Answers.interruptTutorial = false;
        this.Close();
    }
}

}

using System;
using System.Collections.Generic;
using System.ComponentModel;

```

```

using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class DiscreteLogarithmCalculationForm : Form
    {
        public DiscreteLogarithmCalculationForm()
        {
            InitializeComponent();
        }

        private void DoMagicBtn_Click(object sender, EventArgs e)
        {
            BSGSProgressBar.Value = 0;
            BigInteger a, b, m, answer;
            a = 0;
            b = 0;
            m = 0;
            answer = 0;

            ResultLabel.Visible = false;
            ResultLabel.ForeColor = Color.Red;

            BigInteger.TryParse(ABox.Text, out a);
            BigInteger.TryParse(BBox.Text, out b);
            BigInteger.TryParse(MBox.Text, out m);

            if (a < 2 || b < 0 || m < 3 || b > m)
                ResultLabel.Text = "Неправильный ввод данных";
            else if (!(CryptoMath.isPrime(m)))
                ResultLabel.Text = "М должно быть простым";
            else
            {
                if (m.ToString().Length > 10)
                {

```

```

        ResultLabel.Text = "Слишком большое М - \nВозможна
        нехватка памяти";
        ResultLabel.Visible = true;
    }
    try
    {
        BSGSProgressBar.Visible = true;
        this.Enabled = false;
        answer = BabyStepGiantStep.bsgs2(a, b, m, BSGSProgressBar
        );
    }
    catch (OutOfMemoryException)
    {
        ResultLabel.Text = "Нехватка оперативной памяти";
        ResultLabel.Visible = true;
        this.Enabled = true;
        return;
    }
    if (answer == -1)
        ResultLabel.Text = "Такого X не найдено";
    else
    {
        ResultLabel.ForeColor = Color.Green;
        ResultLabel.Text = $"X = {answer}";
        if (BigInteger.ModPow(a, answer, m) != b)
            ResultLabel.Text += "?";
    }
}
BSGSProgressBar.Visible = false;
ResultLabel.Visible = true;
this.Enabled = true;
}

private void PohligButton_Click(object sender, EventArgs e)
{
    BSGSProgressBar.Value = 0;
    BigInteger a, b, m, answer;
    a = 0;
    b = 0;
    m = 0;
    answer = 0;

```

```

ResultLabel.Visible = false;
ResultLabel.ForeColor = Color.Red;

BigInteger.TryParse(ABox.Text, out a);
BigInteger.TryParse(BBox.Text, out b);
BigInteger.TryParse(MBox.Text, out m);

if (a < 2 || b < 0 || m < 3 || b > m)
    ResultLabel.Text = "Неправильный ввод данных";
else if (!(CryptoMath.isPrime(m)))
    ResultLabel.Text = "М должно быть простым";
else
{
    if (m.ToString().Length > 5)
    {
        ResultLabel.Text = "Слишком большое М - \nВозможно крайне
        долгое время работы";
        ResultLabel.Visible = true;
    }
    try
    {
        BSGSProgressBar.Visible = true;
        this.Enabled = false;
        answer = SilverPohligHellman.PohligHellman(a, b, m,
            BSGSProgressBar);
    }
    catch (OutOfMemoryException)
    {
        ResultLabel.Text = "Нехватка оперативной памяти";
        ResultLabel.Visible = true;
        this.Enabled = true;
        return;
    }
    if (answer == -1)
        ResultLabel.Text = "Такого X не найдено";
    else
    {
        ResultLabel.ForeColor = Color.Green;
        ResultLabel.Text = $"X = {answer}";
    }
}

```



```

        if (BigInteger.ModPow(a, answer, m) != b)
        {
            ResultLabel.ForeColor = Color.Red;
            ResultLabel.Text = "Такого X не найдено";
        }
        // ResultLabel.Text += "?";
    }
}
BSGSProgressBar.Visible = false;
ResultLabel.Visible = true;
this.Enabled = true;
}

private void RhoButton_Click(object sender, EventArgs e)
{
    BSGSProgressBar.Value = 0;
    BigInteger a, b, m, answer;
    a = 0;
    b = 0;
    m = 0;
    answer = 0;

    ResultLabel.Visible = false;
    ResultLabel.ForeColor = Color.Red;

    BigInteger.TryParse(ABox.Text, out a);
    BigInteger.TryParse(BBox.Text, out b);
    BigInteger.TryParse(MBox.Text, out m);

    if (a < 2 || b < 0 || m < 3 || b > m)
        ResultLabel.Text = "Неправильный ввод данных";
    else if (!(CryptoMath.isPrime(m)))
        ResultLabel.Text = "М должно быть простым";
    else if (!(CryptoMath.isPrime((m - 1) / 2)))
        ResultLabel.Text = "Порядок группы М должен быть простым";
    else
    {
        if (m.ToString().Length > 5)
        {
            ResultLabel.Text = "Слишком большое М - \nВозможно долгое
                время работы";
        }
    }
}

```

```

        ResultLabel.Visible = true;
    }
    try
    {
        BSGSProgressBar.Visible = true;
        this.Enabled = false;
        answer = Pollards_Rho.Pollard(a, b, m, BSGSProgressBar);
    }
    catch (OutOfMemoryException)
    {
        ResultLabel.Text = "Нехватка оперативной памяти";
        ResultLabel.Visible = true;
        this.Enabled = true;
        return;
    }
    if (answer == -1)
        ResultLabel.Text = "Такого X не найдено";
    else
    {
        ResultLabel.ForeColor = Color.Green;
        ResultLabel.Text = $"X = {answer}";

        if (BigInteger.ModPow(a, answer, m) != b)
        {
            ResultLabel.ForeColor = Color.Red;
            ResultLabel.Text = "Такого X не найдено";
        }
        //ResultLabel.Text += "?";
    }
}
BSGSProgressBar.Visible = false;
ResultLabel.Visible = true;
this.Enabled = true;
}

private void RandomDataButton_Click(object sender, EventArgs e)
{
    Random rand = new Random();
    int size = rand.Next(11) + 2;
    BigInteger M = 0;
    while (!CryptoMath.isPrime((M-1)/2))

```

```

        M = CryptoMath.genSimpleRand(size);
        MBox.Text = M.ToString();
        ABox.Text = (CryptoMath.genRand(rand.Next(size - 1) + 1)+2).
            ToString();
        BBox.Text = (CryptoMath.genRand(rand.Next(size - 1) + 1)).
            ToString();
    }
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Numerics;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace ElgamalTutor

```

```

{
    public partial class DiscrLogTutor : Form
    {
        static Random r = new Random();
        static BigInteger base1, base2, num1, num2, mod1, mod2, ans1,
            ans2;
        public DiscrLogTutor()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
            textBox2.Select(0, 0);
            base1 = new BigInteger(r.Next(3)+2);
            base2 = new BigInteger(r.Next(4)+2);
            mod1 = base1;
            mod2 = base2;
            BigInteger rnd1 = new BigInteger(r.Next(5) + 3);
            BigInteger rnd2 = new BigInteger(r.Next(7) + 3);
            num1 = BigInteger.ModPow(base1, rnd1, mod1);
            num2 = BigInteger.ModPow(base2, rnd2, mod2);

```

```

while (mod1 <= base1 || num1 == base1 || num1 == base1*base1
    || num1 == 0)
{
    mod1 = new BigInteger(r.Next(5) + 2);
    rnd1 = new BigInteger(r.Next(5) + 3);
    num1 = BigInteger.ModPow(base1, rnd1, mod1);
}
while (mod2 <= base2 || num2 == base2 || num2 == base2 *
    base2 || num2 == 0)
{
    mod2 = new BigInteger(r.Next(7) + 2);
    rnd2 = new BigInteger(r.Next(5) + 3);
    num2 = BigInteger.ModPow(base1, rnd2, mod2);
}

Task1Label.Text =
    "Попробуйте найти логарифм по основанию " + base1.ToString
        () + " и модулю " + mod1.ToString() + " от числа " +
        num1.ToString() + ":";
Task2Label.Text =
    "Логарифм по основанию " + base2.ToString() + " и модулю "
        + mod2.ToString() + " от числа " + num2.ToString() +
        ":";
}

private void ReturnBtn_Click(object sender, EventArgs e)
{
    Answers.interruptTutorial = false;
    Answers.formToShow -= 2;
    this.Close();
}

private void nextBtn_Click(object sender, EventArgs e)
{
    if (BigInteger.TryParse(AnswerBox1.Text, out ans1) &&
        BigInteger.TryParse(AnswerBox2.Text, out ans2))
    {
        Answers.DiscreteLogAnswers[0] = (num1 == BigInteger.ModPow(
            base1, ans1, mod1));
        Answers.DiscreteLogAnswers[1] = (num2 == BigInteger.ModPow(

```

```

        base2, ans2, mod2));
//var newForm = new TutorForm3();
//newForm.Show();
Answers.interruptTutorial = false;
this.Close();
    }
}
}
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class ElGamalInfoForm : Form
    {
        public ElGamalInfoForm()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }

        private void endBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            this.Close();
        }
    }
}

```

```

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class EncryptForm : Form
    {
        private BigInteger p = new BigInteger(0);
        private BigInteger g = new BigInteger(0);
        private BigInteger y = new BigInteger(0);
        private BigInteger k = new BigInteger(0);
        private BigInteger a = new BigInteger(0);
        private BigInteger b = new BigInteger(0);
        private BigInteger M = new BigInteger(0);
        public EncryptForm()
        {
            InitializeComponent();
        }

        private void EncryptBtn_Click(object sender, EventArgs e)
        {
            if (M < p)
            {
                a = BigInteger.ModPow(g, k, p);
                aBox.Text = a.ToString();
                b = BigInteger.ModPow(y, k, p);
                b *= M;
                b %= p;
                bBox.Text = b.ToString();
            }
        }
    }
}

```

```

    }
}

private void pBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(pBox.Text, out p);
}

private void gBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(gBox.Text, out g);
}

private void kBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(kBox.Text, out k);
    if (k > p - 1) k = 0;
}

private void MBox_TextChanged(object sender, EventArgs e)
{
    Encoding encoding = Encoding.GetEncoding(1251);
    byte[] converted = encoding.GetBytes(MBox.Text + (char)0);
    M = new BigInteger(converted);
    if (M > p)
        ERRORLabel.Visible = true;
    else
        ERRORLabel.Visible = false;
}

private void genKBtn_Click(object sender, EventArgs e)
{
    // matan.DIGITS = 10;
    do
    {
        k = CryptoMath.genRand(10);
    } while (k > p - 1);
    kBox.Text = k.ToString();
}

private void yBox_TextChanged(object sender, EventArgs e)

```

```

        {
            BigInteger.TryParse(yBox.Text, out y);
        }

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class GelfondShanksAlgorithmForm : Form
    {
        public GelfondShanksAlgorithmForm()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }

        private void endBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            this.Close();
        }
    }
}

```



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class GelfondShanksInfoForm : Form
    {
        public GelfondShanksInfoForm()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }

        private void endBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            this.Close();
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class GelfondShanksTheoryForm : Form
    {
        public GelfondShanksTheoryForm()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }

        private void endBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            this.Close();
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace ElgamalTutor

```

```

{
public partial class MainMenuForm : Form
{
    public MainMenuForm()
    {
        InitializeComponent();
    }

    private void startTutorBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        Answers.formToShow = 0;
        var formList = new List<Form>();

        formList.Add(new TutorIntroForm());
        formList.Add(new TutorForm1());
        formList.Add(new TutorForm2());
        formList.Add(new TutorForm3());
        formList.Add(new ElGamalInfoForm());
        formList.Add(new TutorForm4());
        formList.Add(new TutorForm5());
        formList.Add(new TutorForm6());
        formList.Add(new TutorForm7());
        formList.Add(new DiscrLogTutor());
        formList.Add(new DiscreteLogAlgorithmsInfoForm());
        formList.Add(new GelfondShanksInfoForm());
        formList.Add(new GelfondShanksTheoryForm());
        formList.Add(new GelfondShanksAlgorithmForm());
        formList.Add(new SPHInfoForm());
        formList.Add(new SPHParticularAlgorithmForm());
        formList.Add(new SPHFullAlgorithmForm());
        formList.Add(new RhoInfoForm());
        formList.Add(new TestQuestionsForm1());
        formList.Add(new TestQuestionsForm2());
        formList.Add(new TestQuestionsForm3());
        formList.Add(new TestQuestionsForm4());
        formList.Add(new TestQuestionsForm5());
        formList.Add(new TutorResults());

        while (!Answers.tutorialEnded && !Answers.interruptTutorial)
        {

```

```

        Answers.interruptTutorial = true;
        if (Answers.formToShow >= formList.Count)
            break;
        formList[Answers.formToShow].ShowDialog();
        Answers.formToShow++;
    }
    //this.Close();
}

private void MainMenuForm_Load(object sender, EventArgs e)
{

}

private void fiBtn_Click(object sender, EventArgs e)
{
    var newForm = new CalcfiForm();
    newForm.ShowDialog();
}

private void primitBtn_Click(object sender, EventArgs e)
{
    var newForm = new PrimitiveRootForm();
    newForm.ShowDialog();
}

private void genKeysBtn_Click(object sender, EventArgs e)
{
    var newForm = new DecryptForm();
    newForm.ShowDialog();
}

private void decryptBtn_Click(object sender, EventArgs e)
{
    var newForm = new EncryptForm();
    newForm.ShowDialog();
}

private void primeTestBtn_Click(object sender, EventArgs e)
{
    var newForm = new PrimeTestForm();

```

```

        newForm.ShowDialog();
    }

    private void modPowBtn_Click(object sender, EventArgs e)
    {
        var newForm = new modPowForm();
        newForm.ShowDialog();
    }

    private void AboutBtn_Click(object sender, EventArgs e)
    {
        var newForm = new AboutForm();
        newForm.ShowDialog();
    }

    private void BSGSBtn_Click(object sender, EventArgs e)
    {
        var newForm = new DiscreteLogarithmCalculationForm();
        newForm.ShowDialog();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class modPowForm : Form
    {
        BigInteger X = new BigInteger(0);
        BigInteger Y = new BigInteger(0);
        BigInteger M = new BigInteger(0);
    }
}

```

```

public modPowForm()
{
    InitializeComponent();
}

private void modPowBtn_Click(object sender, EventArgs e)
{
    BigInteger.TryParse(XBox.Text, out X);
    BigInteger.TryParse(YBox.Text, out Y);
    BigInteger.TryParse(MBox.Text, out M);
    if (X*Y*M > 0)
    {
        BigInteger result;
        result = BigInteger.ModPow(X, Y, M);
        modPowBox.Text = result.ToString();
    }
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Numerics;
using System.Windows.Forms;

namespace ElgamalTutor
{
    class Pollards_Rho
    {
        static BigInteger[] ext_euclid(BigInteger a, BigInteger b)
        {
            if (b == 0)
            {
                return new BigInteger[] { a, 1, 0 };
            }
            else
            {
                BigInteger[] arr_results = ext_euclid(b, a % b);

```

```

        BigInteger d = arr_results[0];
        BigInteger xx = arr_results[1];
        BigInteger yy = arr_results[2];
        BigInteger x = yy;
        BigInteger y = xx - (a / b) * yy;
        return new BigInteger[] { d, x, y };
    }
}

static BigInteger inverse(BigInteger a, BigInteger n)
{
    return ext_euclid(a, n)[1];
}

static BigInteger[] PollardStep(BigInteger x, BigInteger a,
    BigInteger b, BigInteger G, BigInteger H, BigInteger P,
    BigInteger Q)
{
    BigInteger sub = x % 3;
    if (sub == 0)
    {
        x = x * G % P;
        a = (a + 1) % Q;
    }

    if (sub == 1)
    {
        x = x * H % P;
        b = (b + 1) % Q;
    }

    if (sub == 2)
    {
        x = x * x % P;
        a = a * 2 % Q;
        b = b * 2 % Q;
    }
    return new BigInteger[] { x, a, b };
}

public static BigInteger Pollard(BigInteger G, BigInteger H,

```

```

    BigInteger P, ProgressBar progress = null)
{
    BigInteger Q = (P - 1) / 2;
    // Console.WriteLine($"Q = {Q}");

    BigInteger x = G * H;
    BigInteger a = 1;
    BigInteger b = 1;

    BigInteger X = x;
    BigInteger A = a;
    BigInteger B = b;

    progress.Increment(progress.Width / 10);

    BigInteger tick = P / 20; // for progressbar
    if (tick == 0)
        tick = 1;

    for (BigInteger i = 1; i < P; i++)
    {
        if (progress != null) // progressbar
            if (i % tick == 0) // progressbar!
            {
                progress.Increment(progress.Width / 20); //
                progress.Update();
                progress.Refresh();
                progress.Invalidate();
            }

        BigInteger[] hedgehog = PollardStep(x, a, b, G, H, P, Q);
        x = hedgehog[0];
        a = hedgehog[1];
        b = hedgehog[2];

        BigInteger[] rabbit = PollardStep(X, A, B, G, H, P, Q);
        X = rabbit[0];
        A = rabbit[1];
        B = rabbit[2];
        rabbit = PollardStep(X, A, B, G, H, P, Q);
        X = rabbit[0];
    }
}

```



```

        A = rabbit[1];
        B = rabbit[2];

        if (x == X)
        {
            break;
        }
    }

    BigInteger nom = a - A;
    BigInteger denom = B - b;
    BigInteger res = (inverse(denom, Q) * nom) % Q;
    if (res >= 0 && BigInteger.ModPow(G, res, P) == H)
        return res;
    else if (BigInteger.ModPow(G, res + Q, P) == H)
        return res + Q;
    else
        return -1;
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class PrimeTestForm : Form
    {
        public PrimeTestForm()
        {
            InitializeComponent();
        }
    }
}

```

```

private void testBtn_Click(object sender, EventArgs e)
{
    this.Enabled = false;
    ansLabel.Visible = false;
    BigInteger testInt;
    BigInteger.TryParse(inputBox.Text, out testInt);
    if (testInt > 0)
    {
        if (CryptoMath.isPrime(testInt))
            ansLabel.Text = "Введённое число - простое";
        else ansLabel.Text = "Введённое число - составное";
    }
    else
        ansLabel.Text = "Введено неправильное значение";
    ansLabel.Visible = true;
    this.Enabled = true;
}

private void inputBox_TextChanged(object sender, EventArgs e)
{
    ansLabel.Visible = false;
}
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class PrimitiveRootForm : Form
    {

```

```

int digits = 10;
BigInteger modulo = new BigInteger(0);
BigInteger primit = new BigInteger(0);

public PrimitiveRootForm()
{
    InitializeComponent();
}

private void generatepBtn_Click(object sender, EventArgs e)
{
    modulo = CryptoMath.genSimpleRand(digits);
    pBox.Text = modulo.ToString();
}

private void pBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(pBox.Text, out modulo);
}

private void trueGenBtn1_Click(object sender, EventArgs e)
{
    if (modulo < 3) pBox.Text = "12345";
    primit = CryptoMath.calculatePrimitiveRoot(modulo);
    trueBox.Text = primit.ToString();
}

private void fakeGenBtn_Click(object sender, EventArgs e)
{
    if (modulo < 3) pBox.Text = "12345";
    primit = CryptoMath.calculateFakePrimitiveRoot(modulo);
    fakeBox.Text = primit.ToString();
}

private void digitsBox_TextChanged(object sender, EventArgs e)
{
    if (int.TryParse(digitsBox.Text, out digits))
        return;
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainMenuForm());
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class RhoInfoForm : Form
    {
        public RhoInfoForm()
        {
            InitializeComponent();
        }
    }
}

```

```

        textBox1.Select(0, 0);
    }

    private void endBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        this.Close();
    }

    private void ReturnBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        Answers.formToShow -= 2;
        this.Close();
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Numerics;
using System.Windows.Forms;

namespace ElgamalTutor
{
    class SilverPohligHellman
    {

        static ProgressBar progress;

        public static List<BigInteger> PrimeFactorization(BigInteger p)
        {
            BigInteger d = 2;
            List<BigInteger> primeFactors = new List<BigInteger>();
            if (d*d <= p)
                if (p % d == 0)
                {
                    primeFactors.Add(d);

```

```

        p /= d;
    }
    d += 1;
    while (d * d <= p)
    {
        while (p % d == 0)
        {
            primeFactors.Add(d);
            p /= d;
        }
        d += 2;
        progress.Increment(progress.Width / 100);
    }
    if (p > 1)
        primeFactors.Add(p);
    return primeFactors;
}

public static List<BigInteger[]> CountOccurences(List<
    BigInteger> primeFactors)
{
    List<BigInteger[]> res = new List<BigInteger[]>();
    var set_of_unique = primeFactors.Select(value => value).
        Distinct();
    foreach (var x in set_of_unique)
    {
        progress.Increment(progress.Width / 100);
        res.Add(new BigInteger[] { x, primeFactors.Count(elem =>
            elem == x) });
    }
    return res;
}

public static BigInteger[] ExtendedGCD(BigInteger a, BigInteger
    b)
{
    BigInteger a2 = 1;
    BigInteger a1 = 0;
    BigInteger b1 = 1;
    BigInteger b2 = 0;

```

```

while (b > 0)
{
    BigInteger q = a / b;
    BigInteger r = a % b;
    BigInteger temp = a1;
    a1 = a2 - q * a1;
    a2 = temp;
    temp = b1;
    b1 = b2 - q * b1;
    b2 = temp;
    a = b;
    b = r;
}
return new BigInteger[] { a, a2, b2 };
}

public static BigInteger modularInverse(BigInteger b,
    BigInteger n)
{
    BigInteger[] res = ExtendedGCD(b, n);
    BigInteger g = res[0];
    BigInteger x = res[1];
    BigInteger ret = 0;
    if (g == 1)
    {
        ret = (x % n);
        while (x < 0) x += n;
        return x;
    }
    return -1;
}

public static BigInteger ChineseRemainder(List<BigInteger[]>
    pairs)
{
    BigInteger N = pairs[0][1];
    BigInteger X = 0;
    for (int i = 1; i < pairs.Count; i++)
    {
        // BigInteger[] ni = pairs[i,];
        N *= pairs[i][1];
    }
}

```

```

    progress.Value = 100 * (pairs.Count * 2 / i);
}
for (int i = 0; i < pairs.Count; i++)
{
    progress.Value = (int)(pairs.Count * 2 / (i+1));
    BigInteger ai = pairs[i][0];
    BigInteger ni = pairs[i][1];
    BigInteger mi = (N / ni);
    X += mi * ai * ExtendedGCD(mi, ni)[1];
}
while (X < 0) X += N;
return X % N;
}

public static BigInteger ShanksAlgorithm(BigInteger alpha,
    BigInteger beta, BigInteger n)
{
    return BabyStepGiantStep.bsgs2(alpha, beta, n);
    BigInteger m = (n-1).Sqrt() + 1;
    BigInteger a = BigInteger.ModPow(alpha, m, n);
    BigInteger b = ExtendedGCD(alpha, n)[1];
    List<BigInteger[]> L1 = new List<BigInteger[]>();
    List<BigInteger[]> L2 = new List<BigInteger[]>();

    for (int ii = 0; ii < m; ii++)
    {
        L1.Add(new BigInteger[] { ii, BigInteger.ModPow(a, ii, n)
            });
        BigInteger second = beta * BigInteger.Pow(b, ii) % n;
        while (second < 0) second += n;
        L2.Add(new BigInteger[] { ii, second });
    }

    L1 = L1.OrderBy(x => x[1]).ToList();
    L2 = L2.OrderBy(x => x[1]).ToList();

    int i = 0;
    int j = 0;
    bool found = false;
    while (!found && i < m && j < m)
    {

```



```

        if (L1[j][1] == L2[i][1])
            return m * L1[j][0] + L2[i][0] % n;
        else if (BigInteger.Abs(L1[j][1]) > BigInteger.Abs(L2[i][1]))
            i += 1;
        else
            j += 1;
    }
    return -1;
}

public static BigInteger[] CongruencePair(BigInteger g,
    BigInteger h, BigInteger p, BigInteger q, BigInteger e,
    BigInteger e1, BigInteger e2)
{
    BigInteger alphaInverse = modularInverse(e1, p);
    BigInteger x = 0;
    for (BigInteger i = 1; i < e+1; i++)
    {
        progress.Value = 100 * (int)(i / e+1);
        BigInteger a = BigInteger.ModPow(e1, q.Power(e - 1), p);
        BigInteger b = BigInteger.ModPow(e2 * alphaInverse.Power(x)
            , q.Power(e - i), p);
        x += ShanksAlgorithm(a, b, p) * (q.Power(i - 1));
    }
    return new BigInteger[] { x, q.Power(e) };
}

public static BigInteger PohligHellman(BigInteger h, BigInteger
    g, BigInteger p, ProgressBar prg = null)
{
    progress = prg;
    List<BigInteger[]> CountOccurencesList = CountOccurences(
        PrimeFactorization(p - 1));

    BigInteger tick = CountOccurencesList.Count / 100; // for
        progressbar
    if (tick == 0)
        tick = 1;

    var CongruenceList = new List<BigInteger[]>();

```

```

for (int i = 0; i < CountOccurencesList.Count; i++)
{
    progress.Value = 100 * (i+1) / (CountOccurencesList.Count
        +1);
    progress.Update();
    progress.Refresh();
    progress.Invalidate();
    BigInteger e1 = (h.Power((p - 1) / (CountOccurencesList[i]
        ][0].Power(CountOccurencesList[i][1])))) % p;
    BigInteger e2 = (g.Power((p - 1) / (CountOccurencesList[i]
        ][0].Power(CountOccurencesList[i][1])))) % p;
    try
    {
        CongruenceList.Add(CongruencePair(g, h, p,
            CountOccurencesList[i][0], CountOccurencesList[i][1],
            e1, e2));
        // progress.Increment(upd_value);
    }
    catch(Exception e)
    {
        return -1;
    }
}
return ChineseRemainder(CongruenceList);
}
////////////////////////////////////
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace ElgamalTutor
{

```

```

public partial class SPHFullAlgorithmForm : Form
{
    public SPHFullAlgorithmForm()
    {
        InitializeComponent();
        textBox1.Select(0, 0);
    }

    private void ReturnBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        Answers.formToShow -= 2;
        this.Close();
    }

    private void endBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        this.Close();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class SPHInfoForm : Form
    {
        public SPHInfoForm()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }
    }
}

```

```

    }

    private void ReturnBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        Answers.formToShow -= 2;
        this.Close();
    }

    private void endBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        this.Close();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class SPHParticularAlgorithmForm : Form
    {
        public SPHParticularAlgorithmForm()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
        }
    }
}

```

```

        this.Close();
    }

    private void endBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        this.Close();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class TestQuestionsForm1 : Form
    {
        public TestQuestionsForm1()
        {
            InitializeComponent();
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }

        private void nextBtn_Click(object sender, EventArgs e)
        {
            //Сюда прикрутить проверку ответов
            Answers.TestAnswers[0] = TrueAnswer1RButton.Checked;

```

```

        Answers.TestAnswers[1] = TrueAnswer2RButton.Checked;
        Answers.interruptTutorial = false;
        this.Close();
    }

    private void TestQuestionsForm1_Load(object sender, EventArgs e)
    {

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class TestQuestionsForm2 : Form
    {
        public TestQuestionsForm2()
        {
            InitializeComponent();
        }

        private void nextBtn_Click(object sender, EventArgs e)
        {
            Answers.TestAnswers[2] = TrueAnswer3RButton.Checked;
            Answers.TestAnswers[3] = TrueAnswer4RButton.Checked;
            Answers.interruptTutorial = false;
            this.Close();
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {

```

```

        Answers.interruptTutorial = false;
        Answers.formToShow -= 2;
        this.Close();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class TestQuestionsForm3 : Form
    {
        public TestQuestionsForm3()
        {
            InitializeComponent();
        }

        private void nextBtn_Click(object sender, EventArgs e)
        {
            Answers.TestAnswers[4] = TrueAnswer5RButton.Checked;
            Answers.TestAnswers[5] = TrueAnswer6RButton.Checked;
            Answers.interruptTutorial = false;
            this.Close();
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }
    }
}

```

```

}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class TestQuestionsForm4 : Form
    {
        public TestQuestionsForm4()
        {
            InitializeComponent();
        }

        private void nextBtn_Click(object sender, EventArgs e)
        {
            Answers.TestAnswers[6] = TrueAnswer3RButton.Checked;
            Answers.TestAnswers[7] = TrueAnswer4RButton.Checked;
            Answers.interruptTutorial = false;
            this.Close();
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;

```



```

using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class TestQuestionsForm5 : Form
    {
        public TestQuestionsForm5()
        {
            InitializeComponent();
        }

        private void nextBtn_Click(object sender, EventArgs e)
        {
            Answers.TestAnswers[8] = TrueAnswer3RButton.Checked;
            Answers.TestAnswers[9] = TrueAnswer4RButton.Checked;
            Answers.interruptTutorial = false;
            this.Close();
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class TutorForm1 : Form
    {

        int g1, x1, m1, answer1;
        int g2, x2, m2, answer2;
        int g3, x3, m3, answer3;

        Random rand = new Random();

        public TutorForm1()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
            g1 = rand.Next(2, 4);
            x1 = rand.Next(2, 4);
            m1 = rand.Next(3, 20);
            QuestionLabel1.Text =
                "Попробуйте возвести " + g1.ToString() + " в степень " + x1
                .ToString() + " по модулю " + m1.ToString();
            g2 = rand.Next(2, 10);
            x2 = rand.Next(2, 6);
            m2 = rand.Next(5, 20);
            QuestionLabel2.Text =
                g2.ToString() + "^" + x2.ToString() + " mod " + m2.ToString
                () + " = ";
            g3 = rand.Next(2, 10);
            x3 = rand.Next(2, 6);
            m3 = rand.Next(5, 20);
            QuestionLabel3.Text =
                g3.ToString() + "^" + x3.ToString() + " mod " + m3.ToString
                () + " = ";
        }

        private void nextBtn_Click(object sender, EventArgs e)
        {
            if (int.TryParse(answerBox1.Text, out answer1) && int.

```

```

        TryParse(answerBox2.Text, out answer2) && int.TryParse(
            answerBox3.Text, out answer3))
    {
        Answers.modpowAnswers[0] = (answer1 == BigInteger.ModPow(g1
            , x1, m1));
        Answers.modpowAnswers[1] = (answer2 == BigInteger.ModPow(g2
            , x2, m2));
        Answers.modpowAnswers[2] = (answer3 == BigInteger.ModPow(g3
            , x3, m3));
        //var newForm = new TutorForm2();
        Answers.interruptTutorial = false;
        this.Close();
        //newForm.Show();
    }
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}

private void TutorForm1_Load(object sender, EventArgs e)
{

}

private void answerBox_TextChanged(object sender, EventArgs e)
{
}
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class TutorForm2 : Form
    {
        BigInteger fi1, fi2, fi3;
        BigInteger ans1, ans2, ans3;

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }

        Random rand = new Random();
        public TutorForm2()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
            fi1 = rand.Next(2, 30);
            QuestionLabel1.Text = "fi(" + fi1 + ") = ";
            fi2 = rand.Next(2, 30);
            if (fi2 == fi1) fi2++;
            QuestionLabel2.Text = "fi(" + fi2 + ") = ";
            fi3 = rand.Next(2, 30);
            if (fi3 == fi1) fi3++;
            if (fi3 == fi2) fi3++;
            QuestionLabel3.Text = "fi(" + fi3 + ") = ";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (BigInteger.TryParse(answerBox1.Text, out ans1) &&
                BigInteger.TryParse(answerBox2.Text, out ans2) &&
                BigInteger.TryParse(answerBox3.Text, out ans3))
            {
                Answers.EulerAnswers[0] = (ans1 == CryptoMath.phi(fi1));
                Answers.EulerAnswers[1] = (ans2 == CryptoMath.phi(fi2));
            }
        }
    }
}

```

```

        Answers.EulerAnswers[2] = (ans3 == CryptoMath.phi(fi3));
        //var newForm = new TutorForm3();
        //newForm.Show();
        Answers.interruptTutorial = false;
        this.Close();
    }
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

```

```

namespace ElgamalTutor

```

```

{
    public partial class TutorForm3 : Form
    {
        BigInteger ans1, ans2;
        public TutorForm3()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }

        private void ReturnBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            Answers.formToShow -= 2;
            this.Close();
        }

        private void TutorForm3_Load(object sender, EventArgs e)
        {

```

```

    }

    private void nextBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        BigInteger.TryParse(answerBox1.Text, out ans1);
        BigInteger.TryParse(answerBox2.Text, out ans2);
        Answers.ReverseAnswers[0] = ans1 == 7;
        Answers.ReverseAnswers[1] = ans2 == 21;
        //var newForm = new TutorForm4();
        //newForm.Show();
        this.Close();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElgamalTutor
{
    public partial class TutorForm4 : Form
    {
        public TutorForm4()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }

        private void nextBtn_Click(object sender, EventArgs e)
        {
            Answers.interruptTutorial = false;
            //var newForm = new TutorForm5();

```

```

        //newForm.Show();
        this.Close();
    }

    private void ReturnBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        Answers.formToShow -= 2;
        this.Close();
    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class TutorForm5 : Form
    {
        private BigInteger p = new BigInteger(0);
        private BigInteger g = new BigInteger(0);
        private BigInteger x = new BigInteger(0);
        private BigInteger y = new BigInteger(0);
        private long digits = 30;
        public TutorForm5()
        {
            digits = 30;

```

```

    InitializeComponent();
}

//генерация p
private void generatepBtn_Click(object sender, EventArgs e)
{
    p = CryptoMath.genSimpleRand(digits);
    gtextBox.Text = "";
    xtextBox.Text = "";
    ytextBox.Text = "";
    ptextBox.Text = p.ToString();
}

private void generategBtn_Click(object sender, EventArgs e)
{
    this.Enabled = false;
    if (p>2)
        if (CryptoMath.DIGITS > 10)
            g = CryptoMath.calculateFakePrimitiveRoot(p);
        else
            g = CryptoMath.calculatePrimitiveRoot(p);
    gtextBox.Text = g.ToString();
    this.Enabled = true;
}

private void generatexBtn_Click(object sender, EventArgs e)
{
    if (p>2)
        do
            x = CryptoMath.genRand(digits);
        while (x == 0 || x >= p);
    xtextBox.Text = x.ToString();
}

private void generateyBtn_Click(object sender, EventArgs e)
{
    if (p > 2)
        y = BigInteger.ModPow(g, x, p);
    ytextBox.Text = y.ToString();
}

```



```

private void nextBtn_Click(object sender, EventArgs e)
{
    if (CryptoMath.isPrime(p) && p > 2 && x > 1 && g > 1 && y >
        0)
    {
        //var newForm = new TutorForm6(g, p, y, x);
        //newForm.Show();
        Answers.g = g;
        Answers.p = p;
        Answers.y = y;
        Answers.x = x;
        Answers.interruptTutorial = false;
        this.Close();
    }
    if (!CryptoMath.isPrime(p)) wrongpLabel.Visible = true;
}

private void ptextBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(ptextBox.Text, out p);
}

private void gtextBox_TextChanged(object sender, EventArgs e)
{
}

private void TutorForm2_Load(object sender, EventArgs e)
{
}

private void ptextBox_TextChanged_1(object sender, EventArgs e)
{
    BigInteger.TryParse(ptextBox.Text, out p);
}

private void xtextBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(xtextBox.Text, out x);
}

```

```

    }

    private void ytextBox_TextChanged(object sender, EventArgs e)
    {
        // ytextBox.Text = BigInteger.ModPow(g, x, p).ToString();
    }

    private void ReturnBtn_Click(object sender, EventArgs e)
    {
        Answers.interruptTutorial = false;
        Answers.formToShow -= 2;
        this.Close();
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class TutorForm6 : Form
    {
        private bool hasLoadedKey = false;
        private BigInteger p = new BigInteger(0);
        private BigInteger g = new BigInteger(0);
        private BigInteger x = new BigInteger(0);
        private BigInteger y = new BigInteger(0);
        private BigInteger k = new BigInteger(0);
        private BigInteger a = new BigInteger(0);
        private BigInteger b = new BigInteger(0);
        private BigInteger M = new BigInteger(0);
        public TutorForm6(/*BigInteger garg, BigInteger parg,

```

```

        BigInteger yarg, BigInteger xarg*/)
{
    InitializeComponent();
    //g = Answers.g;
    //p = Answers.p;
    //y = Answers.y;
    //x = Answers.x;
    //gLabel.Text += g.ToString();
    //pLabel.Text += p.ToString();
    //yLabel.Text += y.ToString();
}

private void generatekBtn_Click(object sender, EventArgs e)
{
    do
    {
        k = CryptoMath.genRand(20);
    } while (k > p - 1);
    kBox.Text = k.ToString();
}

private void calcaBtn_Click(object sender, EventArgs e)
{
    a = BigInteger.ModPow(g, k, p);
    aBox.Text = a.ToString();
}

private void MBox_TextChanged(object sender, EventArgs e)
{
    Encoding encoding = Encoding.GetEncoding(1251);
    byte[] converted = encoding.GetBytes(MBox.Text + (char)0);
    M = new BigInteger(converted);
    if (M >= p) wrongMLabel.Visible = true;
    else wrongMLabel.Visible = false;
    aBox.Text = "";
    bBox.Text = "";
    a = 0;
    b = 0;
}

```

```

private void calcbbtn_Click(object sender, EventArgs e)
{
    //подозрительный код, может взять и не заработать
    b = BigInteger.ModPow(y, k, p);
    b *= M;
    b %= p;
    bBox.Text = b.ToString();
}

private void nextBtn_Click(object sender, EventArgs e)
{
    if (M < p && M > 0 && a > 0 && b > 0 && k > 0)
    {
        //var newForm = new TutorForm7(g, p, y, x, a, b);
        //newForm.Show();
        Answers.interruptTutorial = false;
        Answers.a = a;
        Answers.b = b;
        Answers.g = g;
        Answers.p = p;
        Answers.y = y;
        Answers.x = x;
        this.Close();
    }
}

private void label9_Click(object sender, EventArgs e)
{
}

private void kBox_TextChanged(object sender, EventArgs e)
{
    BigInteger.TryParse(kBox.Text, out k);
}

private void TutorForm6_Load(object sender, EventArgs e)
{
    if (!hasLoadedKey)
    {
        g = Answers.g;
        p = Answers.p;
    }
}

```

```

        y = Answers.y;
        x = Answers.x;
        gLabel.Text += g.ToString();
        pLabel.Text += p.ToString();
        yLabel.Text += y.ToString();
        hasLoadedKey = true;
    }
}

private void ReturnBtn_Click(object sender, EventArgs e)
{
    Answers.interruptTutorial = false;
    Answers.formToShow -= 2;
    this.Close();
}
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;

namespace ElgamalTutor
{
    public partial class TutorForm7 : Form
    {
        private BigInteger p = new BigInteger(0);
        private BigInteger g = new BigInteger(0);
        private BigInteger x = new BigInteger(0);
        private BigInteger y = new BigInteger(0);
        private BigInteger a = new BigInteger(0);
        private BigInteger b = new BigInteger(0);
        private BigInteger M = new BigInteger(65);
        public TutorForm7(/*BigInteger garg, BigInteger parg,
```

```

        BigInteger yarg, BigInteger xarg, BigInteger aarg,
        BigInteger barg/*, BigInteger Marg*/)
{
    InitializeComponent();
    //g = Answers.a;
    //p = Answers.b;
    //y = Answers.y;
    //x = Answers.x;
    //a = Answers.a;
    //b = Answers.b;
    ///M = Marg;
    //gLabel.Text += g.ToString();
    //pLabel.Text += p.ToString();
    //yLabel.Text += y.ToString();
    //xLabel.Text += x.ToString();
    //aLabel.Text += a.ToString();
    //bLabel.Text += b.ToString();
}

private void label2_Click(object sender, EventArgs e)
{

}

private void DecryptBtn_Click(object sender, EventArgs e)
{
    Encoding encoding = Encoding.GetEncoding(1251);
    BigInteger apx = BigInteger.ModPow(a, x, p);
    apx = BigInteger.ModPow(apx, p - 2, p);
    M = (b * apx) % p;
    MNumBox.Text = M.ToString();
    byte[] textBytes = M.ToByteArray();
    MtextBox.Text = encoding.GetString(textBytes);
}

private void endBtn_Click(object sender, EventArgs e)
{
    if (M > 0)
    {
        //var newForm = new TutorResults();
        //newForm.Show();
    }
}

```

```

        Answers.interruptTutorial = false;
        this.Close();
    }
}

private void TutorForm7_Load(object sender, EventArgs e)
{
    g = Answers.g;
    p = Answers.p;
    y = Answers.y;
    x = Answers.x;
    a = Answers.a;
    b = Answers.b;
    //M = Marg;
    gLabel.Text += g.ToString();
    pLabel.Text += p.ToString();
    yLabel.Text += y.ToString();
    xLabel.Text += x.ToString();
    aLabel.Text += a.ToString();
    bLabel.Text += b.ToString();
}

private void ReturnBtn_Click(object sender, EventArgs e)
{
    Answers.interruptTutorial = false;
    Answers.formToShow -= 2;
    this.Close();
}
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace ElgamalTutor
{
    public partial class TutorIntroForm : Form
    {
        public TutorIntroForm()
        {
            InitializeComponent();
            textBox1.Select(0, 0);
        }

        private void nextBtn_Click(object sender, EventArgs e)
        {
            //var newForm = new TutorForm1();
            //newForm.Show();
            Answers.interruptTutorial = false;
            this.Close();
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace ElgamalTutor
{
    public partial class TutorResults : Form
    {
        private static string BoolToResult(bool b)
        {
            if (b) return "Берно";
            else return "Неберно";
        }

        public TutorResults()

```



```

{
    InitializeComponent();
}

private void endBtn_Click(object sender, EventArgs e)
{
    this.Close();
}

private void TutorResults_Load(object sender, EventArgs e)
{
    if (Answers.modpowAnswers[0])
        powmodAnsLabel1.Text += "Верно";
    else
        powmodAnsLabel1.Text += "Неверно";
    if (Answers.modpowAnswers[1])
        powmodAnsLabel2.Text += "Верно";
    else
        powmodAnsLabel2.Text += "Неверно";
    if (Answers.modpowAnswers[2])
        powmodAnsLabel3.Text += "Верно";
    else
        powmodAnsLabel3.Text += "Неверно";
    if (Answers.EulerAnswers[0])
        EulerAnsLabel1.Text += "Верно";
    else
        EulerAnsLabel1.Text += "Неверно";
    if (Answers.EulerAnswers[1])
        EulerAnsLabel2.Text += "Верно";
    else
        EulerAnsLabel2.Text += "Неверно";
    if (Answers.EulerAnswers[2])
        EulerAnsLabel3.Text += "Верно";
    else
        EulerAnsLabel3.Text += "Неверно";
    if (Answers.ReverseAnswers[0])
        ReverseAnsLabel1.Text += "Верно";
    else
        ReverseAnsLabel1.Text += "Неверно";
    if (Answers.ReverseAnswers[1])
        ReverseAnsLabel2.Text += "Верно";

```

```

else
    ReverseAnsLabel2.Text += "Неверно";
if (Answers.DiscreteLogAnswers[0])
    DiscreteLogAnsLabel1.Text += "Верно";
else
    DiscreteLogAnsLabel1.Text += "Неверно";
if (Answers.DiscreteLogAnswers[1])
    DiscreteLogAnsLabel2.Text += "Верно";
else
    DiscreteLogAnsLabel2.Text += "Неверно";
if (Answers.AlgorithmicDLOGAnswers[0])
    AlgAnsLabel1.Text += "Верно";
else
    AlgAnsLabel1.Text += "Неверно";
if (Answers.AlgorithmicDLOGAnswers[1])
    AlgAnsLabel2.Text += "Верно";
else
    AlgAnsLabel2.Text += "Неверно";

////////////////////////////////////

TestAnswer1Label.Text += BoolToResult(Answers.TestAnswers[0])
    ;
TestAnswer2Label.Text += BoolToResult(Answers.TestAnswers[1])
    ;
TestAnswer3Label.Text += BoolToResult(Answers.TestAnswers[2])
    ;
TestAnswer4Label.Text += BoolToResult(Answers.TestAnswers[3])
    ;
TestAnswer5Label.Text += BoolToResult(Answers.TestAnswers[4])
    ;
TestAnswer6Label.Text += BoolToResult(Answers.TestAnswers[5])
    ;
TestAnswer7Label.Text += BoolToResult(Answers.TestAnswers[6])
    ;
TestAnswer8Label.Text += BoolToResult(Answers.TestAnswers[7])
    ;
TestAnswer9Label.Text += BoolToResult(Answers.TestAnswers[8])
    ;
TestAnswer10Label.Text += BoolToResult(Answers.TestAnswers
    [9]);

```

```

//Some LINQ magic
double summa = Answers.modpowAnswers.Sum(a => a ? 1 : 0) +
    Answers.EulerAnswers.Sum(a => a ? 1 : 0)
    + Answers.ReverseAnswers.Sum(a => a ? 1 : 0) + Answers.
        DiscreteLogAnswers.Sum(a => a ? 1 : 0)
    + Answers.TestAnswers.Sum(a => a ? 1 : 0) + Answers.
        AlgorithmicDLOGAnswers.Sum(a => a ? 1 : 0);
double FinalMark = 3 * (summa / 20) + 2;
if (FinalMark % 1 >= 0.5)
    FinalMark += 1;
FinalMarkLabel.Text = ((int)FinalMark).ToString();
}
}
}

```