МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ФИЗИКИ
Кафедра радиофизики

А.В. КАРПОВ, А.Д. АНДРИЯНОВ

КРИПТОГРАФИЧЕСКАЯ СИСТЕМА ЭЛЬ-ГАМАЛЯ

ВИРТУАЛЬНАЯ ЛАБОРАТОРНАЯ РАБОТА

учебно-методическое пособие
по выполнению лабораторных работ

Казань – 2024

УДК 004.056.55 + 003.26 + 004.421.5
ББК 16.84
К26

Печатается по решению учебно-методической комиссии
Института физики
Казанского (Приволжского) федерального университета
(протокол № _09__ от «_29___» ___мая_____ 2024 г.)

Рецензент
кандидат физико-математических наук, доцент кафедры «Информационные технологии и интеллектуальные системы»
Казанского государственного энергетического университета
Р.А. Ишмуратов

Карпов А.В., Андриянов А.Д. КРИПТОГРАФИЧЕСКАЯ СИСТЕМА ЭЛЬГАМАЛЯ. ВИРТУАЛЬНАЯ ЛАБОРАТОРНАЯ РАБОТА. Учебно-методическое пособие по выполнению лабораторных работ для магистрантов и студентов старших курсов. Казань, 2024.30 с.

Учебно-методическое пособие содержит описание виртуальной лабораторной работы по основам криптографии с открытым ключом. Лабораторная работа выполняется на компьютере с установленной на нем программой. Вначале пособия даются краткие теоретические сведения по теме работы. Во второй части приводится описание графического интерфейса программы и порядок выполнения заданий.

Предназначено для студентов, обучающихся по специальности 10.03.01 – Информационная безопасность (Безопасность автоматизированных систем) по дисциплине «Криптографические методы защиты информации» и обучающихся по специальности 03.04.03 – Радиофизика (Распределенные интеллектуальные системы) по дисциплине «Криптографические методы в распределенных интеллектуальных системах», а также может быть рекомендовано при подготовке курсовых/дипломных проектов и магистерских диссертаций.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящее пособие содержит описание виртуальной лабораторной работы по основам криптографии с открытым ключом. Лабораторная работа содержит различные по сложности типы заданий. Большинство заданий имеют базовую сложность. Выполнение этих заданий гарантирует освоение изучаемого материала. Для различных направлений подготовки студентов предусмотрено выполнение заданий повышенной сложности, которые отмечаются знаком (*). Выполнение этих заданий стимулирует углубленное изучение рассматриваемой темы. Пособие заканчивается списком контрольных вопросов, которые позволяют студенту проверить усвоение теории и выделить для себя самое важное.

По результатам выполнения лабораторной работы составляется отчёт, который представляется в виде электронного документа в формате doc или odt. При оформлении отчёта рекомендуется придерживаться приведённой ниже структуры.

Структура отчёта:

1. Фамилия студента, номер учебной группы.
2. Название и номер задания.
3. Цель работы.
4. Краткие теоретические сведения.
5. Описание хода лабораторной работы. По возможности рекомендуется дополнять каждый пункт выполненного задания скриншотами рабочего окна программы, иллюстрирующими выполнение задания.
6. Выводы. Изложение результатов по каждому пункту лабораторного задания должно завершаться подведением основных выводов.

Часть 1. Криптосистемы с открытым ключом. Асимметричное шифрование на основе схемы Эль-Гамаля

Задача дискретного логарифмирования по модулю

Задача дискретного логарифма по модулю простого числа является фундаментальным понятием в теории чисел и криптографии. Эта проблема лежит в основе различных криптографических протоколов, особенно в области криптографии с открытым ключом. Она лежит в основе криптографических систем, таких как обмен ключами Диффи – Хеллмана и алгоритм цифровой подписи (DSA).

Трудность решения задачи дискретного логарифма имеет существенное значение для безопасности этих криптографических схем. Если бы был найден эффективный алгоритм, способный решить задачу дискретного логарифмирования больших чисел, многие распространенные криптографические системы были бы скомпрометированы.

Описание работы алгоритма

Схема Эль-Гамаля (Elgamal) – это криптографическая система с общедоступным ключом, которая была основана на сложности вычисления дискретных логарифмов. Схема Эль-Гамаля является основополагающим компонентом в стандартах цифровой подписи DSA и Государственного стандарта Российской Федерации 34.10-94.

Основателем схемы является Тахер Эль-Гамаль, который разработал её в 1985 году. Он доработал алгоритм Диффи – Хеллмана и в конечном счёте получил два алгоритма, которые и обеспечивают безопасность данной системы. Если сравнивать два алгоритма, RSA и алгоритм Эль-Гамаля, то второй не был запатентован и вследствие чего стал недорогой альтернативой, которая не требовала дополнительных оплат за лицензию. Если рассматривать с юридической точки зрения, то алгоритм попадает под действие патента Диффи – Хелмана.

Рассмотрим работу криптографического алгоритма Эль-Гамаля. Блок-схема алгоритма представлена на рис. 1

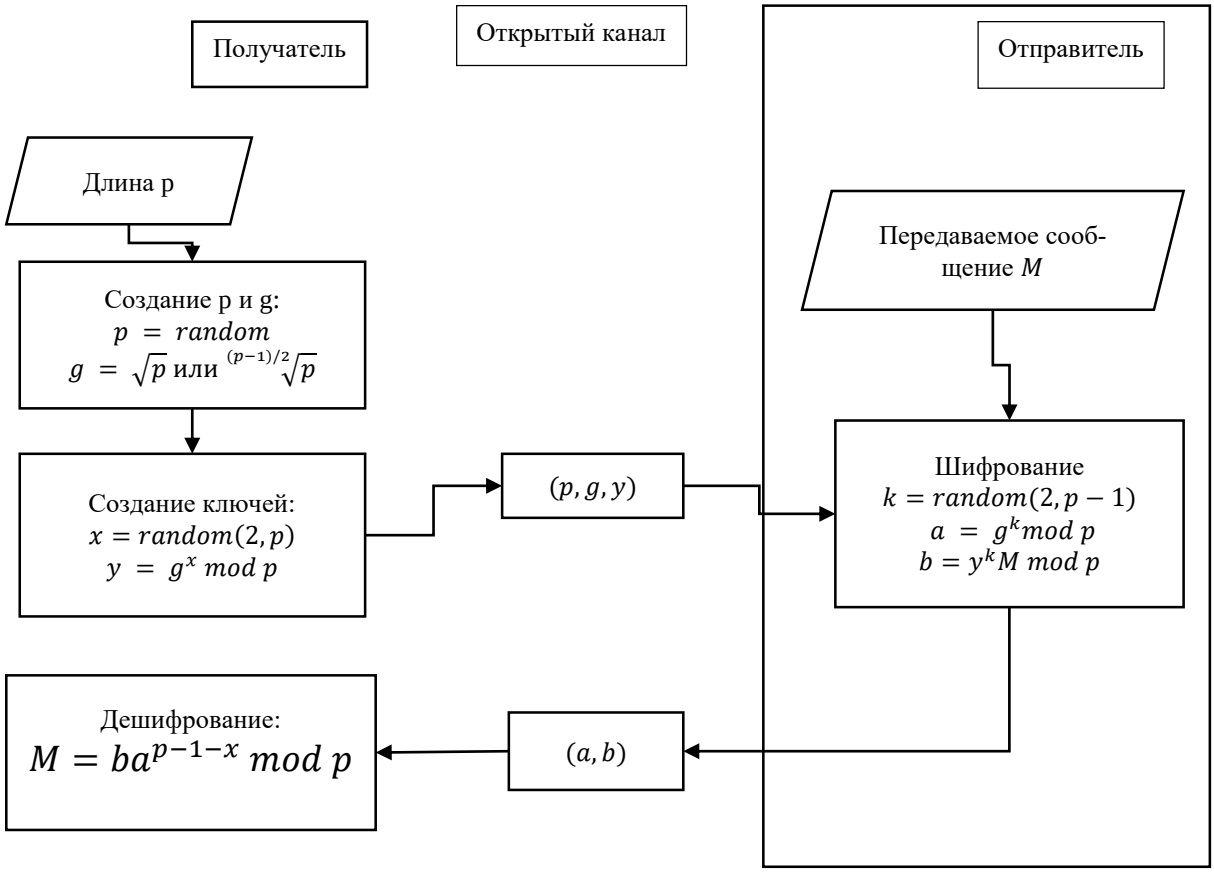


Рис.1. Блок-схема криптографического алгоритма Эль-Гамаля

Процедура генерации ключей:

1. Генерируется произвольное простое число p.
2. Подбирается целое число g – первообразный корень простого числа p.

Первообразный корень g простого числа p такой, что все числа из интервала $[1, p-1]$ могут быть представлены как различные степени числа g по модулю p: $g \bmod p$. Чтобы было просто генерировать такие числа, можно использовать следующий подход: простое число p берётся такое, что $p = 2 \cdot q + 1$, где число q тоже простое. Тогда в качестве g можно взять число, для которого выполняется: $g^q \pmod p \neq 1$ и $1 < g < p - 1$. Такой выбор числа p также и усиливает криптостойкость алгоритма.

3. Подбирается произвольная целочисленная переменная x такая, что $1 < x < p - 1$.
4. Вычисляется $y = g^x mod \ p$.

Общедоступным ключом являются набор y, g, p, закрытым ключом x.

Поскольку открытый ключ известен обеим сторонам и лежит в открытом доступе, то злоумышленник не сможет его подменить, так как это будет заметно всем пользователям системы.

Шифрование

Шифрование происходит в несколько этапов:

1. Подбирается сеансовый ключ k – произвольная целочисленная переменная, взаимно простая с $(p - 1)$, такая, что $1 < k < p - 1$.
2. Происходит вычисление следующих чисел: $a = g^k \, mod \, p$ и $b = y^k \, M \, mod \, p$.

Пара чисел (a, b) является шифротекстом.

Длина шифротекста по схеме Эль-Гамаля будет в два раза больше, чем изначальное сообщение M.

Дешифрование

Законный пользователь, имея информацию о закрытом ключе x, изначальное сообщение может вычислить из шифротекста (a, b) по формуле:

$$M = b(a^x)^{-1} \, mod \, p.$$

Более удобная формула для вычисления исходного текста будет выглядеть так:

$$M = ba^{p-1-x} \, mod \, p.$$

Поскольку в схему Эль-Гамаля нужно ввести произвольную величину k, то данный шифр называют шифром многозначной замены. Если сравнивать схему Эль-Гамаля со схемами с однозначным процессом шифрования, можно отметить, что вероятностный характер шифрования у данной схемы даёт некое преимущество в сфере стойкости системы. Главным недостатком этой схемы является длина зашифрованного текста, потому что он имеет удвоенную длину изначального сообщения.

Криптостойкость алгоритма

Критерий простоты числа, или свойство быть простым числом, играет важную роль в криптографической стойкости шифра Эль-Гамаля. Криптосистема Эль-Гамаля опирается на сложность решения задачи дискретного логарифмирования в конечном поле, которая тесно связана с выбором простых чисел, используемых в процессе шифрования. На практике шифр Эль-Гамаля использует в своих вычислениях большое простое число *p*. Одним из основных подходов доказательства простоты числа *p* является Тест Рабина – Миллера.

Тест Рабина – Миллера представляет собой вероятностный алгоритм, используемый для определения того, является ли данное число простым.

Принцип работы алгоритма следующий:

1. Пусть (*n*) – число, которое требуется проверить на простоту, и (*n* > 2).
2. Приводится (*n* – 1) к виду (2^*sd*), где (*d*) – нечётное число, а (*s*) – степень двойки: (*n* – 1) = (2^*sd*).
3. Для каждого случайно выбранного числа (*a*) проверяются следующие утверждения:
   - $a^d \equiv 1 \pmod{n}$
   - $a^{2^r d} \equiv -1 \pmod{n}$ для всех (*r*) от 0 до (*s* – 1), то (*n*) является составным числом.

Если хотя бы одно условие выполняется, то значит выбранное *a* является свидетелем простоты числа *n*, и число *n* с большой вероятностью может быть простым. Если оба условия не выполняются, то число *n* – составное и дальнейшее тестирование можно прекратить.

Для увеличения вероятности принятия решения, что выбранное число простое, тест повторяется для другого случайно выбранного основания *a.* Если число проходит тест для большого числа случайных оснований, вероятность того, что оно действительно простое, становится очень высокой. Обычно используют от 10 до 50 раундов в зависимости от требуемой надёжности.

Задача дискретного логарифмирования по модулю

Существует три распространенных метода решения задачи дискретного логарифмирования по модулю простого числа:

1. Метод «грубой силы» (перебор) – это простой подход к решению задачи дискретного логарифмирования по модулю простого числа. Вот описание метода грубой силы.

Реализация. Метод грубой силы предполагает систематический перебор всех возможных значений показателя степени для того, чтобы найти правильное решение. Он начинается с начального значения (обычно это 1) и постепенно увеличивает экспоненту до тех пор, пока не будет получен желаемый остаток. Для каждого тестируемого значения экспоненты основание возводится к этому показателю и результат вычисляется по модулю простого модуля.

Эффективность. Хотя метод грубой силы концептуально прост, он крайне неэффективен для больших простых чисел. Количество требуемых вычислений растет экспоненциально с размером простого модуля, что делает его непрактичным для реальных приложений, где задействованы большие числа.

Сложность. Временная сложность метода грубой силы равна $O(p)$, где p – это размер простого модуля. Метод грубой силы практичен только для очень малых простых чисел, где число возможных значений экспоненты поддается управлению. Он может быть использован в образовательных целях или в качестве базового сравнения для более эффективных алгоритмов.

Ограничения. Из-за своей неэффективности метод грубой силы не подходит для практических приложений, связанных с большими числами.

Таким образом, хотя метод грубой силы обеспечивает простой способ решения задачи дискретного логарифма, его неэффективность ограничивает его практическое применение для больших простых чисел. Более сложные алгоритмы, такие как метод «Шаги младенца – шаги гиганта» или алгоритм Полига – Хеллмана, предпочтительны для реальных приложений, где эффективность и безопасность имеют первостепенное значение.

2. Метод «Шаги младенца – шаги гиганта» – это алгоритм, используемый для эффективного решения задачи дискретного логарифмирования по модулю простого числа. Вот описание шагов, задействованных как на этапах «шагов младенца», так и «шагов гиганта»:

Реализация метода.

Нужно решить уравнение $a^x = b \bmod p$, где $a<p$; $b<p$; p – произвольное простое число. Выбираем $m = [\sqrt{p}] + 1$, где $[\sqrt{p}]$ – ближайшее к $\sqrt{p}$ меньшее целое число, (например, $[\sqrt{15}] = 3$).

Далее вычисляем два ряда:

$$\text{Ряд } i\text{: } a^m, a^{2m}, a^{3m}, \ldots\ldots, a^{im} \quad \bmod p; \quad 1 \leq i \leq m$$

$$\text{Ряд } j\text{: } b, ba, ba^2, \ \ldots, ba^j \quad \bmod p; \quad 0 \leq j \leq m-1$$

Ряд i получил название «шаги младенца», а ряд j – «шаги гиганта». Решение получается на том шаге, когда шаги младенца и гиганта совпадают. Значения i и j находим на том шаге, когда выполняется условие $a^{im} = ba^j$, подставляем эти значения в выражение $x = im - j$. находим решения исходного уравнения.

Эффективность метода. Разбивая задачу на два этапа: «шаги младенца – шаги гиганта», мы сокращаем общее количество необходимых вычислений. Метод использует хэш-таблицу или механизм сортировки для эффективного хранения и извлечения промежуточных результатов, снижая общие вычислительные затраты.

Сложность. Временная сложность метода «Шаги младенца – шаги гиганта» равна $O(\sqrt{p})$, что делает его гораздо более эффективным, чем метод грубой силы.

Пригодность. Метод «шаги младенца – шаги гиганта» подходит для умеренно больших простых чисел, где метод грубой силы был бы непрактичен. Он обеспечивает баланс между эффективностью и простотой, что делает его широко используемым алгоритмом в криптографических приложениях.

3**. Алгоритм Полига** – **Хеллмана** – это усовершенствованный метод, используемый для решения задачи дискретного логарифмирования по модулю простого числа, особенно когда простой модуль имеет малые простые множители. Ниже приведен краткий обзор алгоритма Полига – Хеллмана.

Концепция. Алгоритм Полига – Хеллмана предназначен для эффективного решения задачи дискретного логарифмирования путем реализации факторизации модуля, точнее не самого модуля, а числа $n=p-1$. В данном случае задачу разбивают на более мелкие подзадачи, основанные на простых множителях модуля, решая каждую подзадачу независимо и объединяя результаты для нахождения общего решения.

Пусть G – группа порядка $n=p-1$, и $n=$ $p_1^{\alpha_1} p_2^{\alpha_2} \ldots . p_k^{\alpha_k}$ каноническое разложение на простые сомножители.

Например, модуль $p=37$, $n = p-1=36 =2^2 \cdot 3^2$

Если $x=log_g a \ mod n$, то вычислив $x_i = \log_g a \ \ mod\ p_i^{a_i}$ для $1 \le i \le k$, можно восстановить x, используя китайскую теорему о б остатках.

Для того чтобы вычислить x_i, вычисляют коэффициенты $l_0, l_1, \ldots, l_{a_{k-1}}$ в p_i-ичном представлении числа x_i: $x_i=l_0+l_1p_i+$ $+$ $l_{\alpha_k-1} p_i^{\alpha_k-1}$

где $0 \le l_j \le p_i-1$.

Представим метод Полига – Хеллмана следующим алгоритмом:

Алгоритм Полига – Хеллмана:

А). Входные данные: g – порождающий элемент циклической группы порядка n, $a \in G$.

Находим каноническое разложение $n= p_1^{\alpha_1} p_2^{\alpha_2} \ldots . p_k^{\alpha_k}$

В). Для i от 1 до k выполняем следующие действия:

В1. Задаем $q=p_i$, $\alpha=\alpha_i$.

В2. Задаем $\gamma=1$, $l_{-1}=0$.

В3. Вычисляем $\overline{g} = g^{\frac{n}{q}}$

В4. Для j от 1 до $\alpha-1$ выполняем:

4.1. Вычисляем:

$\gamma = \gamma \cdot g^{l_{j-1}q^{j-1}}$ и $\bar{a} = (a\gamma^{-1})q^{\frac{n}{j+1}}$

В4.2. Вычисляем $l_i = \log_g \bar{a}$, (например, используя алгоритм «шаг младенца – шаг великана» или прямой поиск).

В5. Вычисляем $x_i = l_0 + l_1 q + \ldots + l_{\alpha-1} q^{\alpha-1}$.

С. Используя Китайскую теорему об остатках, решаем систему уравнений:

$x = x_i mod p^{a_i}$

И получаем решение: $x = log_g a \ mod \ n$.

Эффективность. Алгоритм Полига–Хеллмана очень эффективен для простых модулей с малыми простыми множителями, так как он уменьшает сложность задачи, разбивая ее на более мелкие, более управляемые подзадачи. Это позволяет проводить параллельные вычисления подзадач, ускоряя общий процесс решения.

Сложность. Сложность алгоритма Полига–Хеллмана равна $O(\sqrt{p})$, где p – простой модуль.

Пригодность. Алгоритм Полига–Хеллмана особенно подходит для простых модулей с малыми простыми множителями, где другие методы, такие как метод грубой силы или шаги гиганта и младенца, могут быть менее эффективными. Он обычно используется в криптографических приложениях, где простой модуль имеет известную факторизацию.

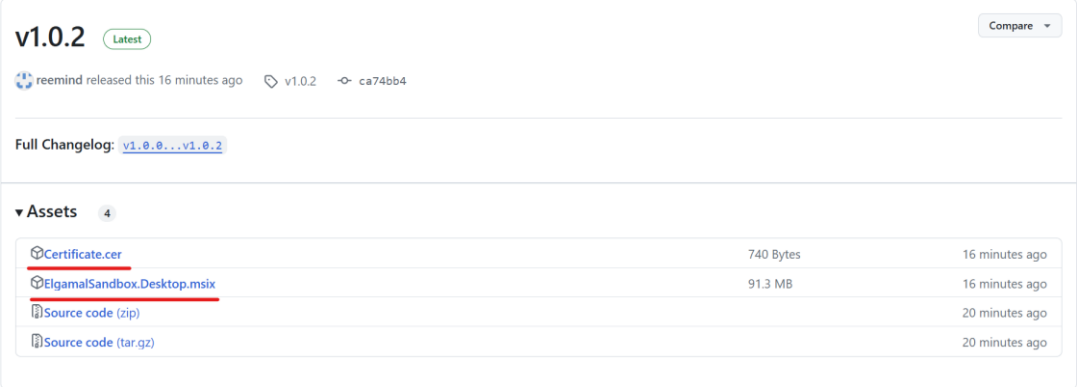
Таким образом, в то время как метод грубой силы прост, но неэффективен, метод шагов младенца – шагов гиганта предлагает более эффективный подход для умеренно больших простых чисел. Алгоритм Полига–Хеллмана отличается своей эффективностью при работе с простыми числами, имеющими малые простые множители. Выбор метода зависит от размера и характеристик используемого простого модуля.

Часть 2. Виртуальная лабораторная работа

Установка программного приложения на компьютер

1. Скачивание установщика и сертификата

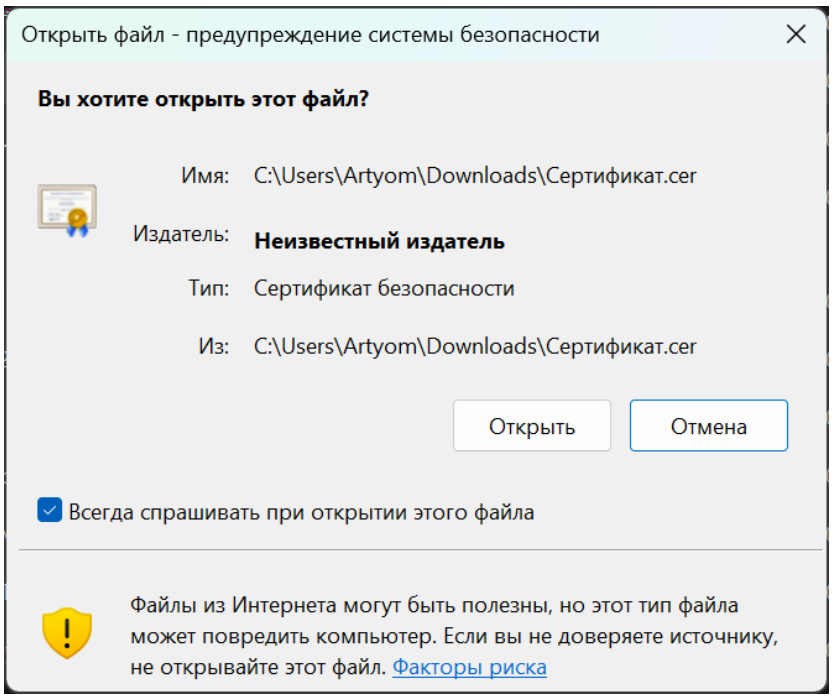
По ссылке https://github.com/reemind/elgamal-sandbox/releases/latest требуется скачать 2 файла: файл сертификата (Certificate.cer) и установщик (ElgamalSandbox.Desktop.msix).



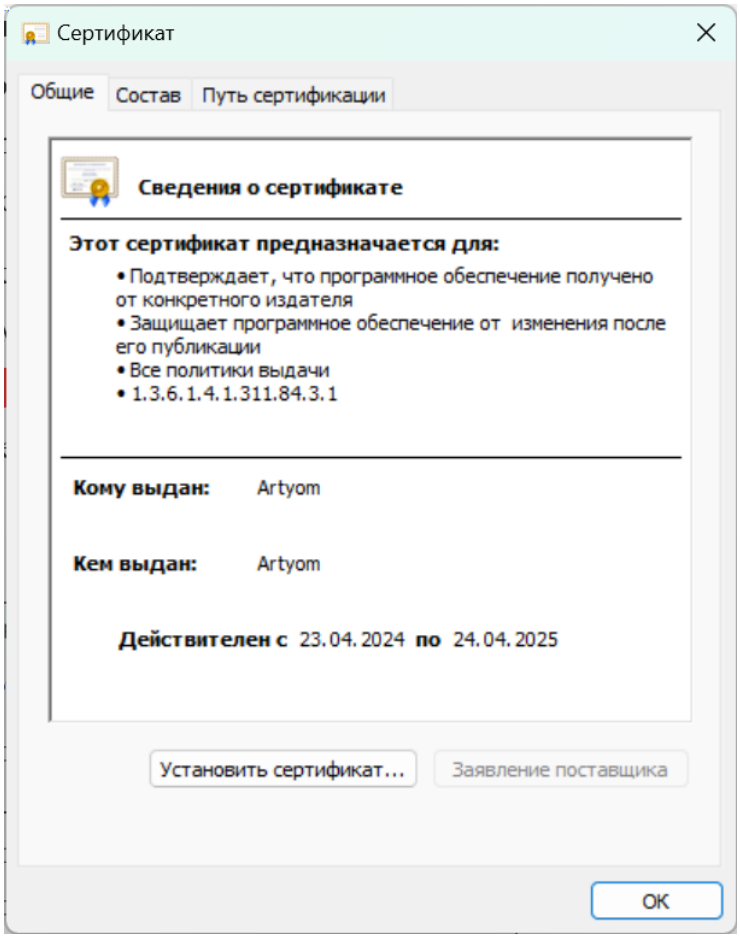
Для установки программного приложения на компьютер пользователя потребуется установить цифровой сертификат и само приложение. Установка сертификата производится один раз. В последствии (для обновления или переустановки приложения) достаточно запустить установщик (см. пункт 3).

2. Установка корневого сертификата в доверенные корневые центры.

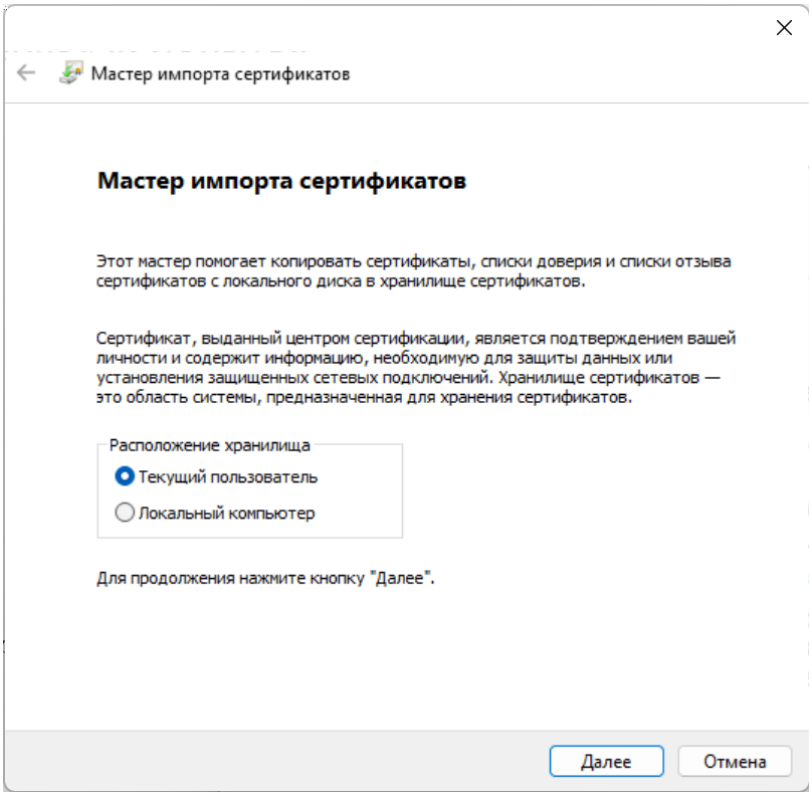
После двойного нажатия на файл сертификата (Certificate.cer), который был скачан на шаге 1, появится окно «Предупреждение системы безопасности», в котором необходимо нажать кнопку «Открыть»:



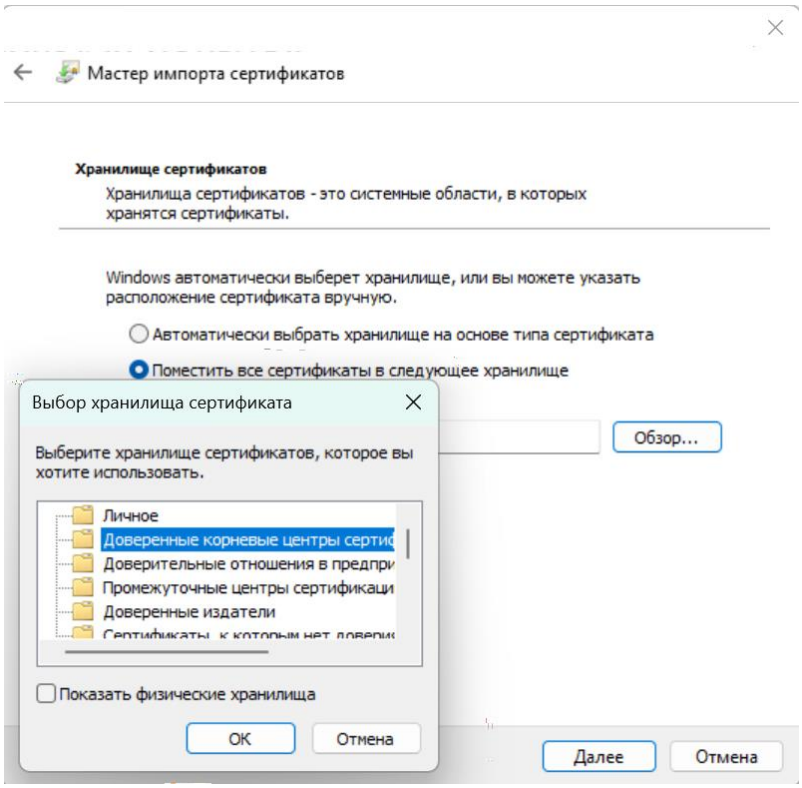
После этого в появившемся новом диалоговом окне следует нажать кнопку "Установить сертификат...":



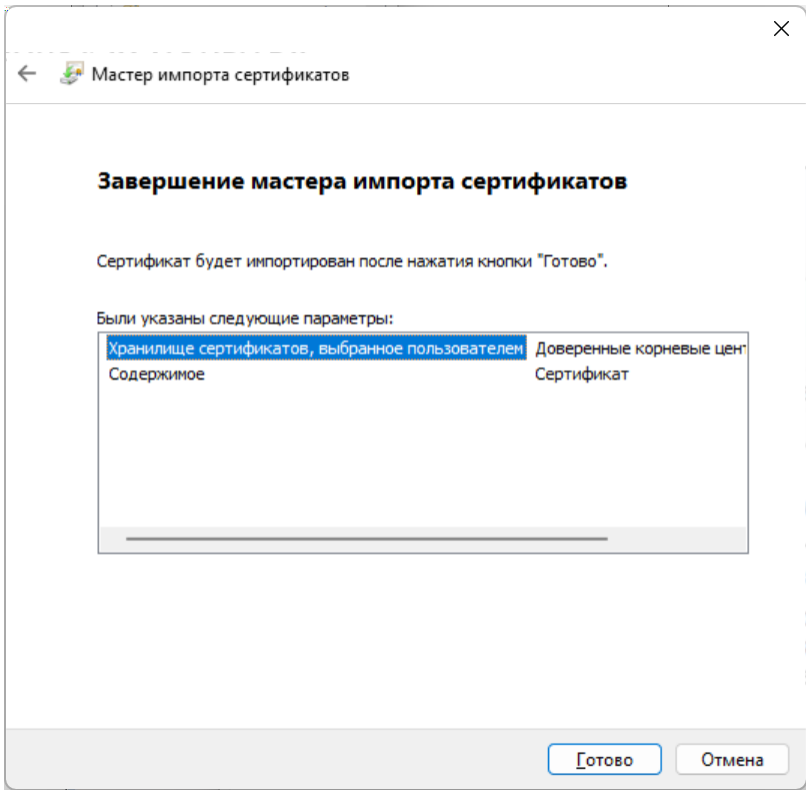
В появившемся окне «Мастер импорта сертификатов» следует нажать кнопку "Далее":



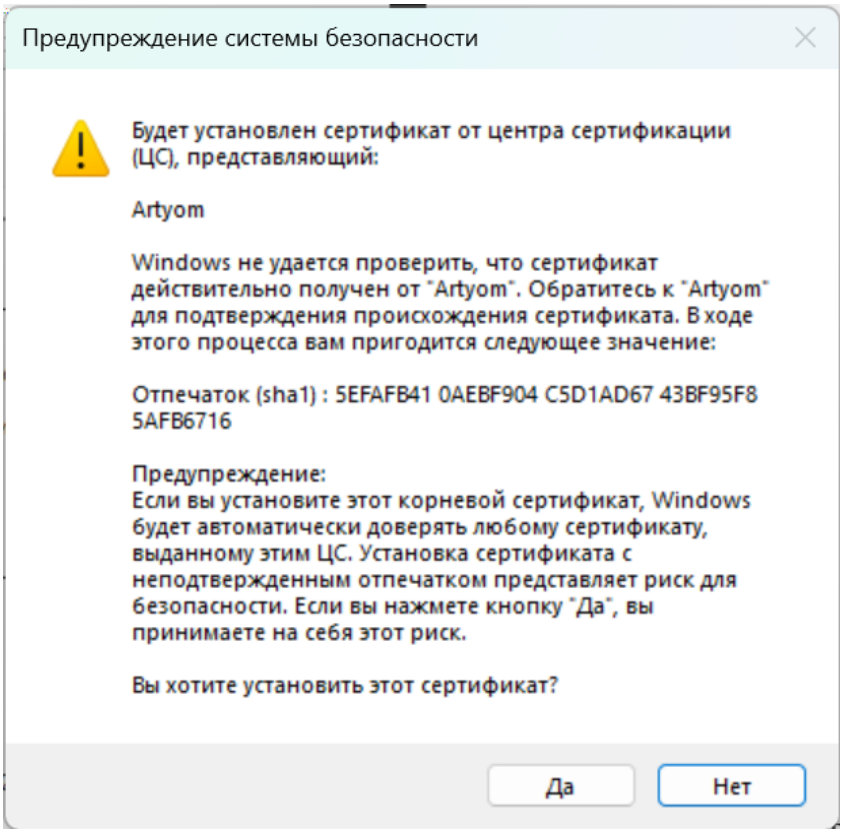
Затем выбрать пункт "Поместить все сертификаты в следующее хранилище" → "Обзор...". В появившемся окне выбрать ветку "Доверенные корневые центры сертификации", нажать "ОК" и «Далее».



В окне «Мастер импорта сертификатов» нажать "Готово":

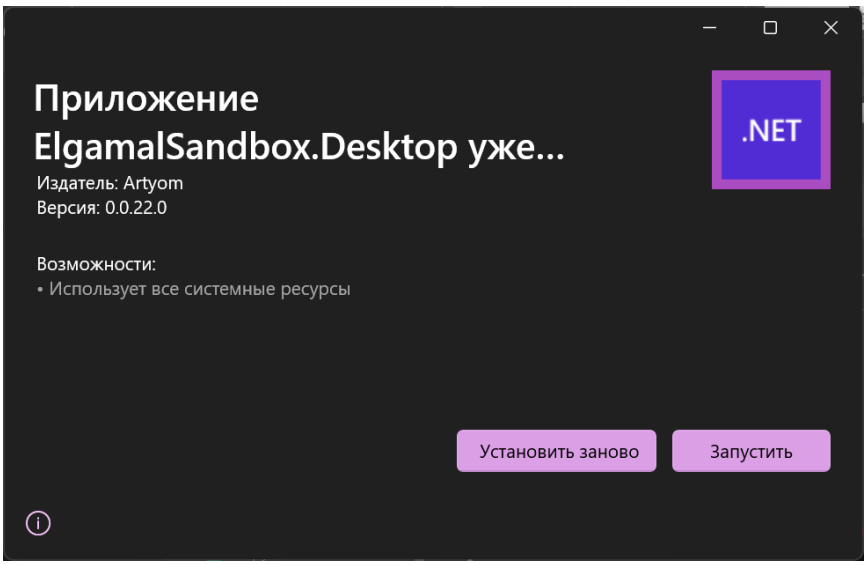


После закрытия окна «Мастер импорта сертификатов» появится окно «Предупреждение о безопасности», в котором следует нажать «Да». В результате проделанных действий сертификат будет успешно установлен.



2. Установка приложения.

Открываем установщик (.msix) и нажимаем «Установить»:

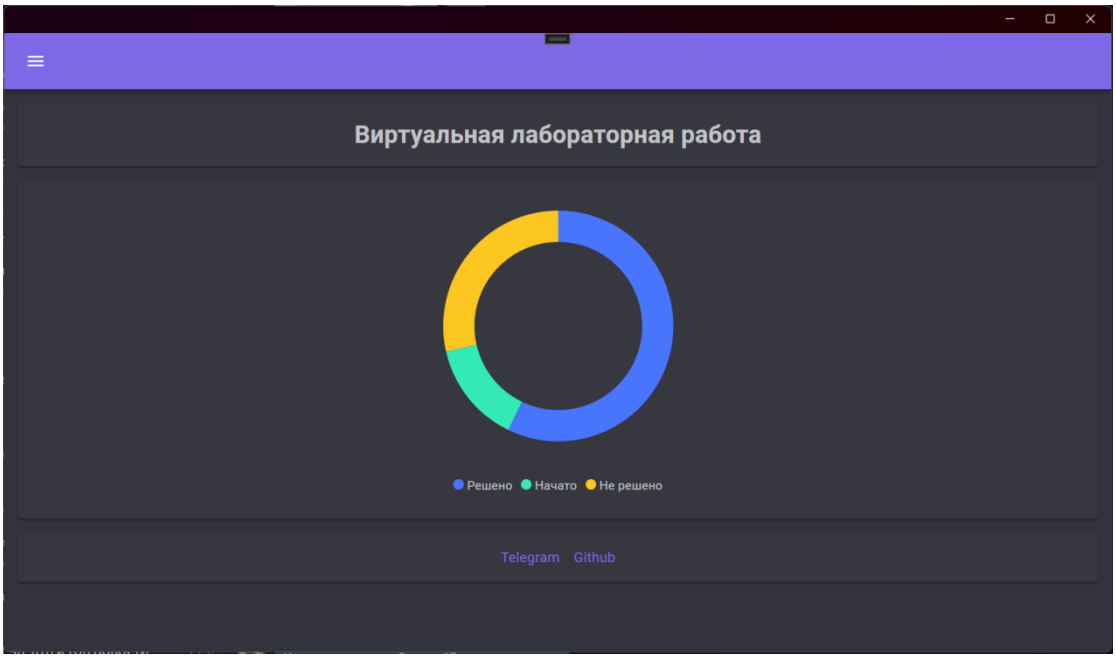


После установки приложение будет доступно как обычно в списке приложений в меню «Пуск».

Описание интерфейса приложения

Главный экран приложения

Главный экран приложения представляет из себя краткую сводку о прогрессе в ходе выполнения задач (заданий) данной лабораторной работы:



Окно задач

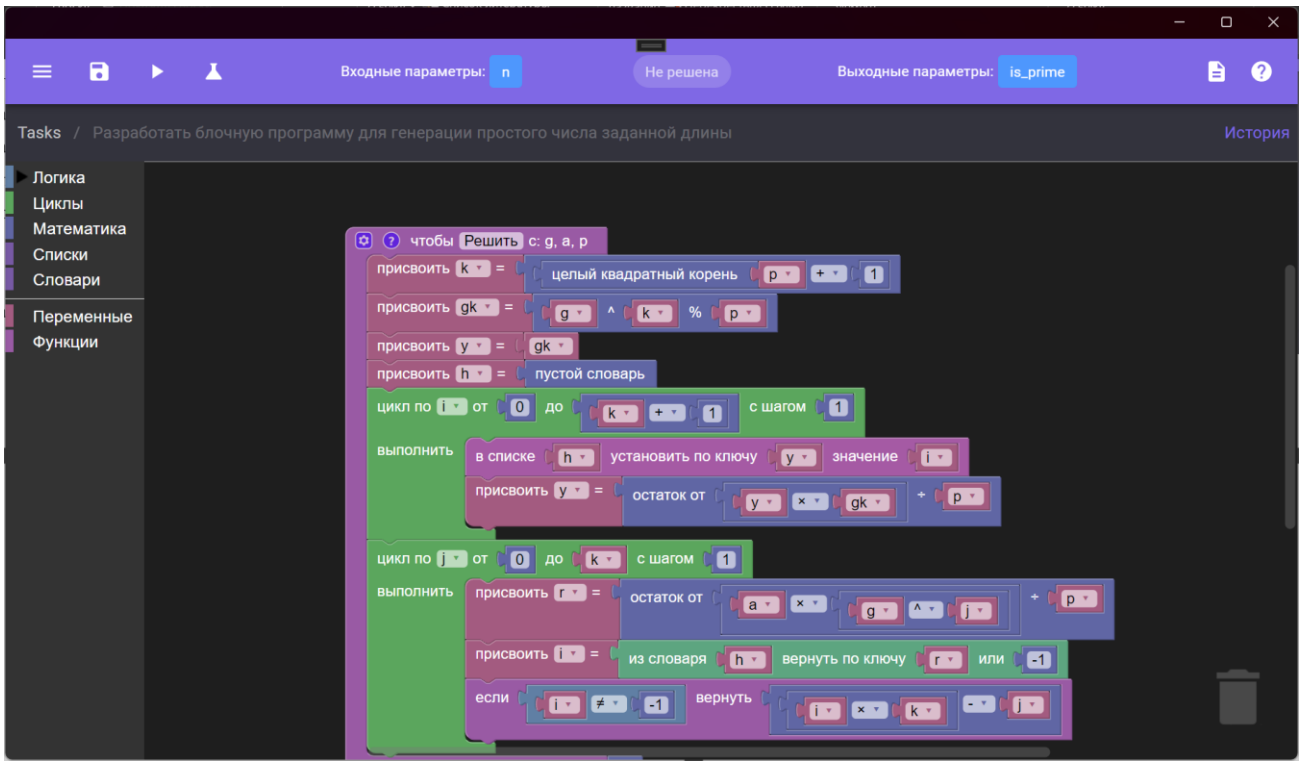
Окно задач представляет из себя список предложенных заданий лабораторной работы. Задания расположены в рекомендованном для выполнения порядке. Кроме порядкового номера и названия задания в таблице отображается количество попыток и время последнего запуска, а также статус решения. Иконка таймера рядом с номером задачи подсказывает, что данная задача может использоваться в тесте эффективности алгоритма. Статус решения определяется наличием хотя бы одного удачного запуска тестов.

Запуск задач и тестов

Окно выполнения задания представляет из себя рабочую область, область инструментария и дополнительных кнопок, расположенные в «шапке» окна. В «шапке» окна отображается дополнительная информация о статусе решения и входных и выходных параметрах. Для решения задачи требуется из готовых блоков составить программу, которая будет записывать на выходную переменную результат работы, исходя из входящих параметров. Существует два варианта запуска алгоритма:

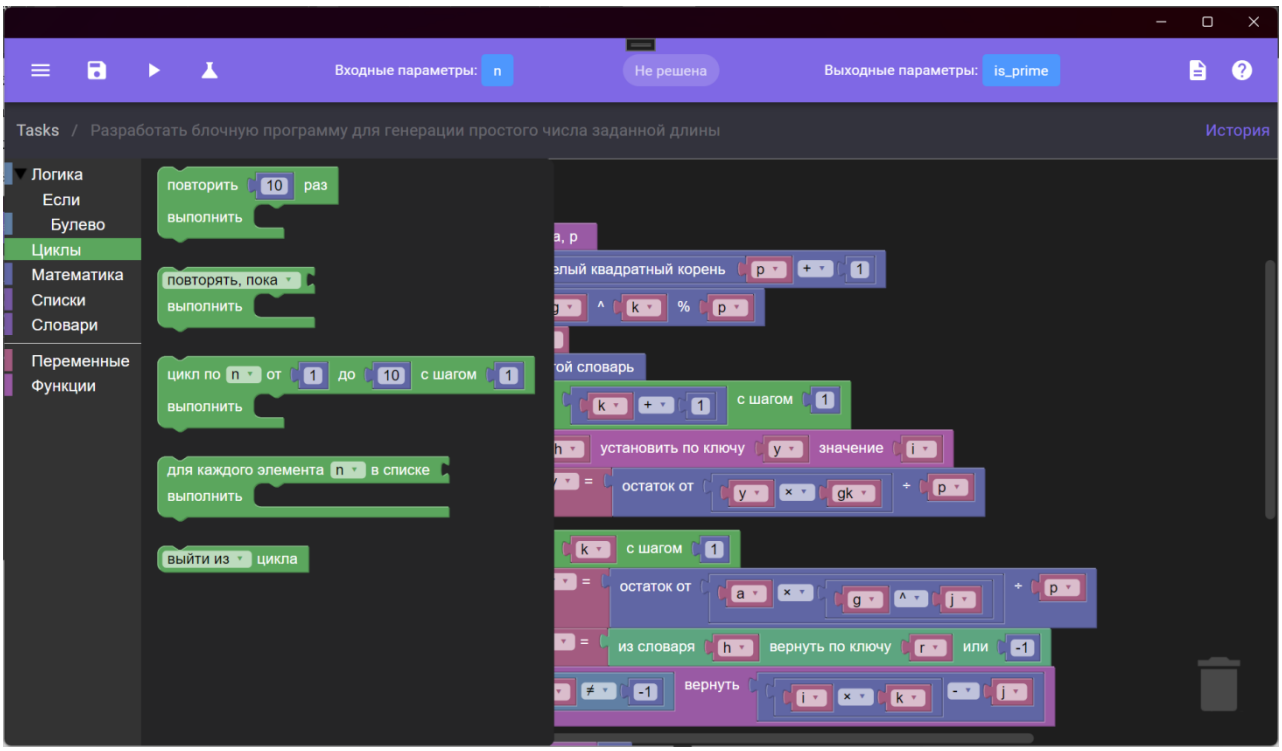
1) ручной запуск программы с входными параметрами, устанавливаемыми вручную;

2) с отображением результатов выполнения и запуск заранее сформированных тестов для данного задания.



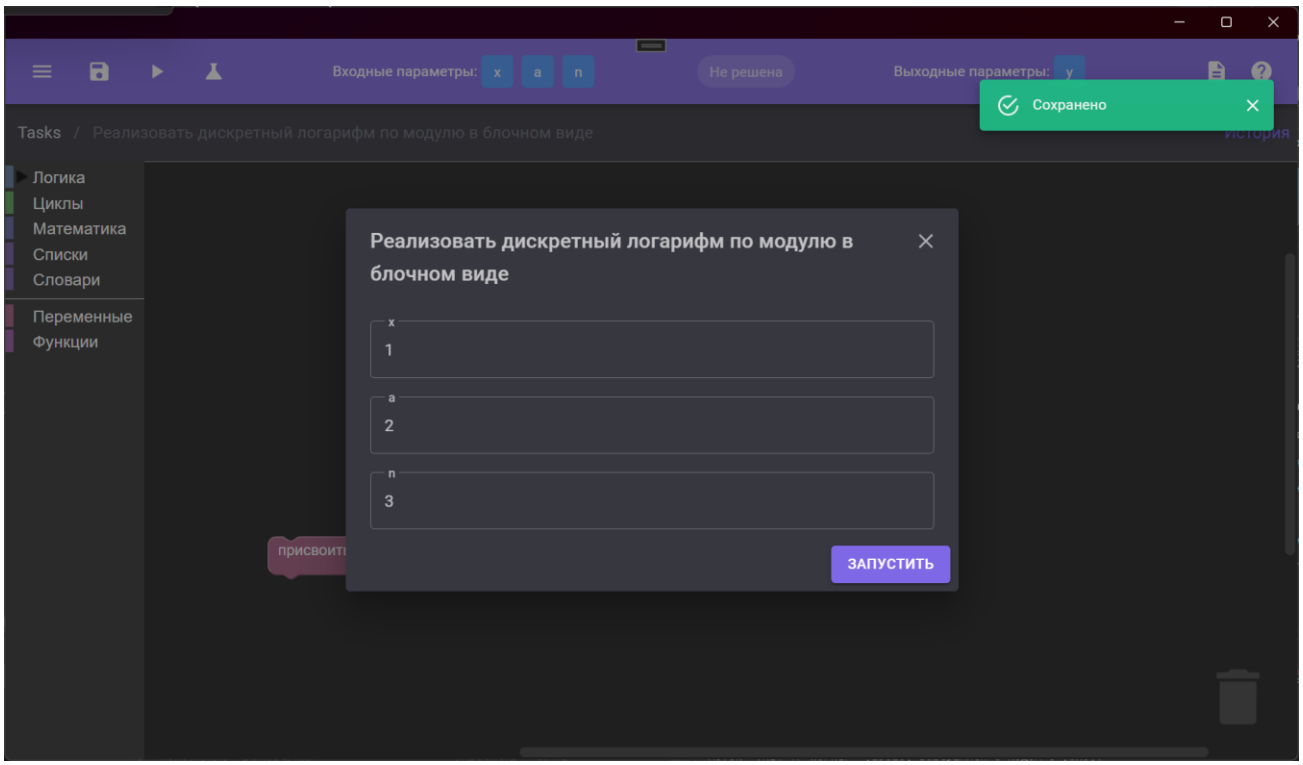
Инструментарий программы представляет из себя несколько категорий готовых блоков. Представлены блоки для работы с булевой логикой, циклами, математика, списками, словарями. Также присутствует функционал для создания, записи и чтения переменных, объявления и использования создаваемых функций.

Блоки размещаются путем нажатия кнопки мышки на нужный блок и последующего перетаскивания на Рабочую область. Блоки можно соединять между собой в последовательные цепочки. Для этого они могут содержать места их соединения с другими блоками. Кроме этого, блоки содержат поля с дополнительно устанавливаемыми параметрами блока.

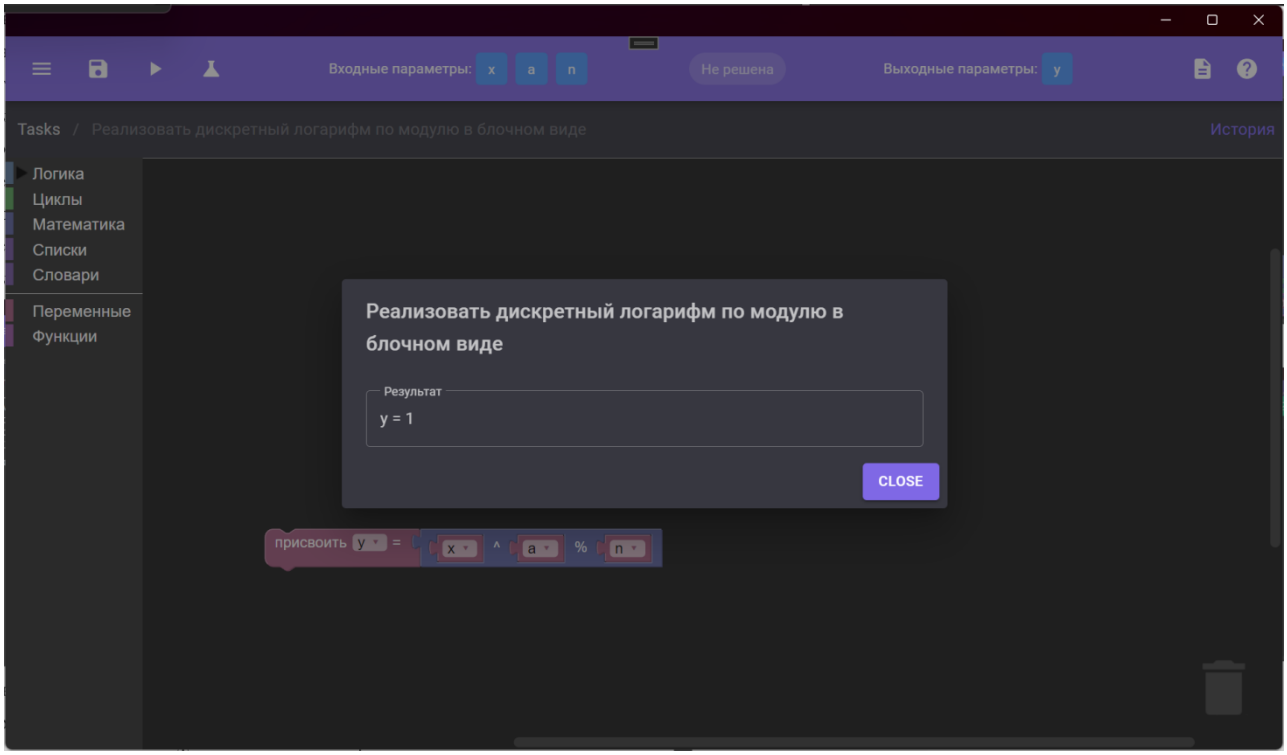


Перед каждым запуском производится сохранение текущей рабочей области. Также сохранение можно произвести вручную путем нажатия на соответствующую иконку.

Для запуска реализации требуется нажать на иконку запуска. Откроется окно установки входных параметров.

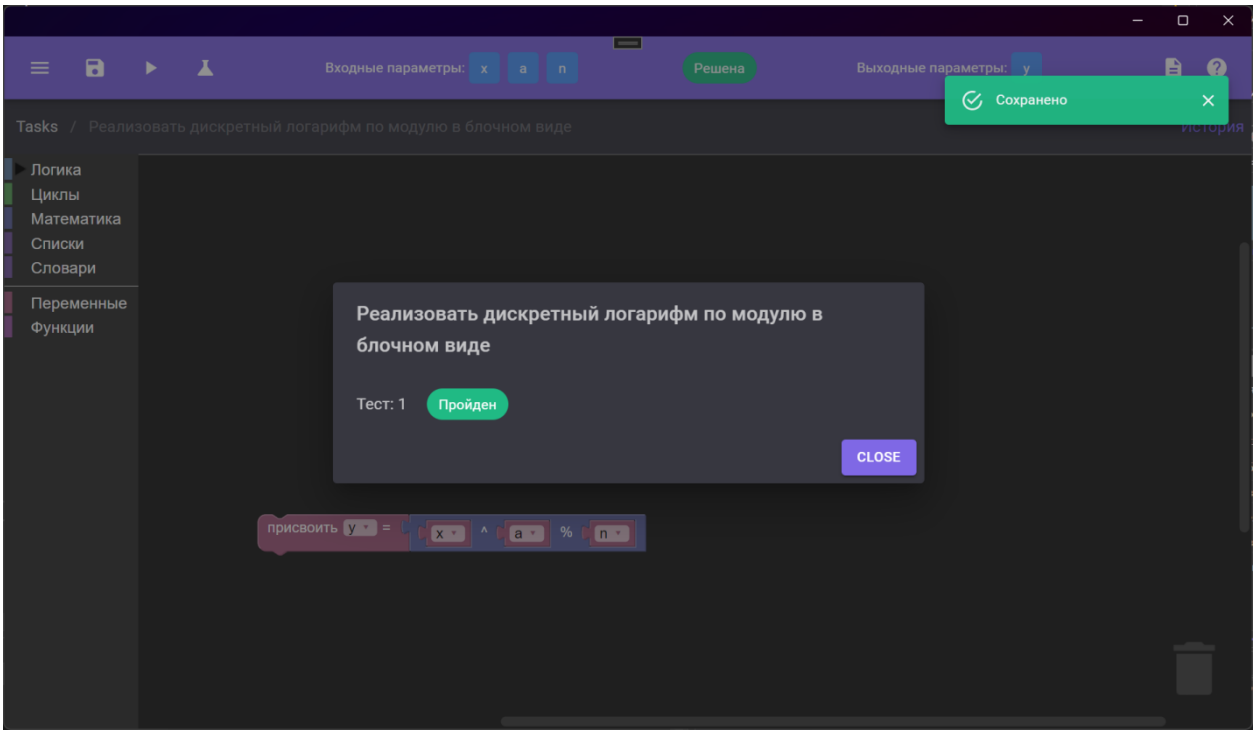


Результат выполнения программы будет отражен в модальном окне.



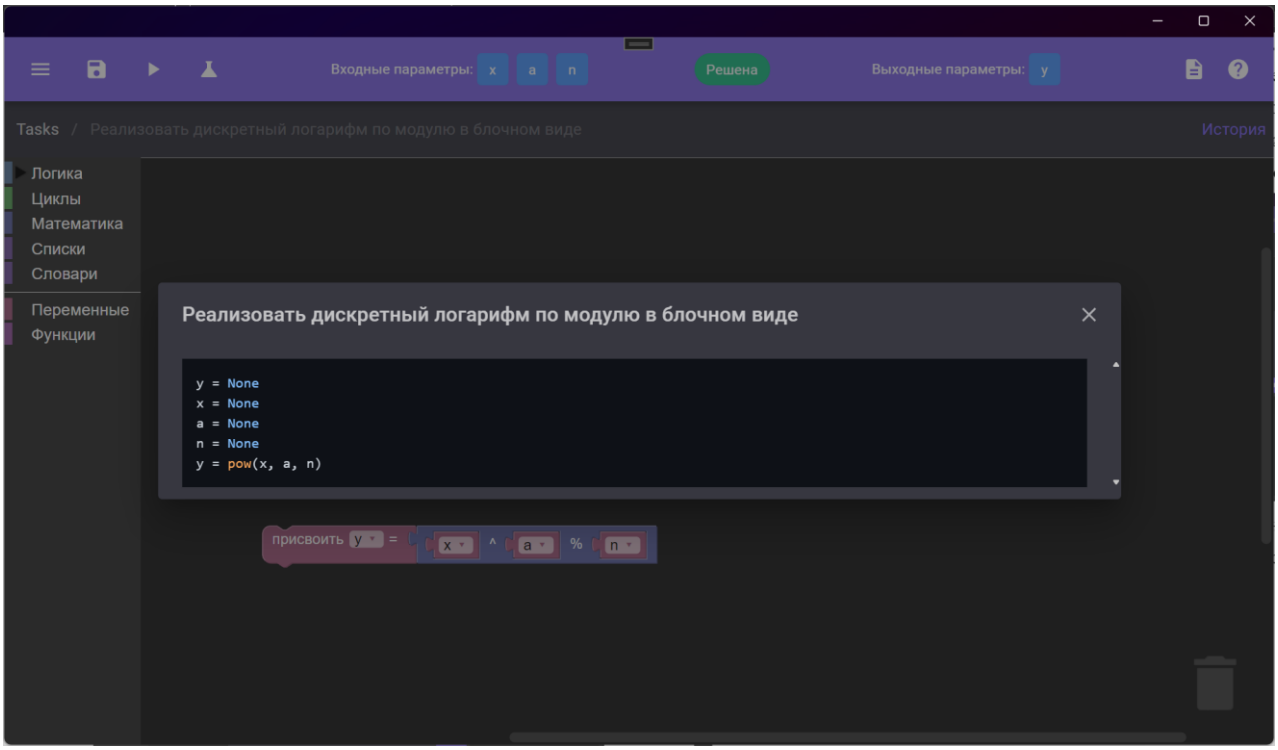
Для запуска тестов требуется нажать на иконку "колбы". После нажатия выполнить хранения теста и запуск всех предопределенных тестов. Смысл тести-

рования заключается в самостоятельной проверке корректности работы программы. Также прохождение тестов важно для заданий, которые используется в измерения эффективности работы алгоритма.

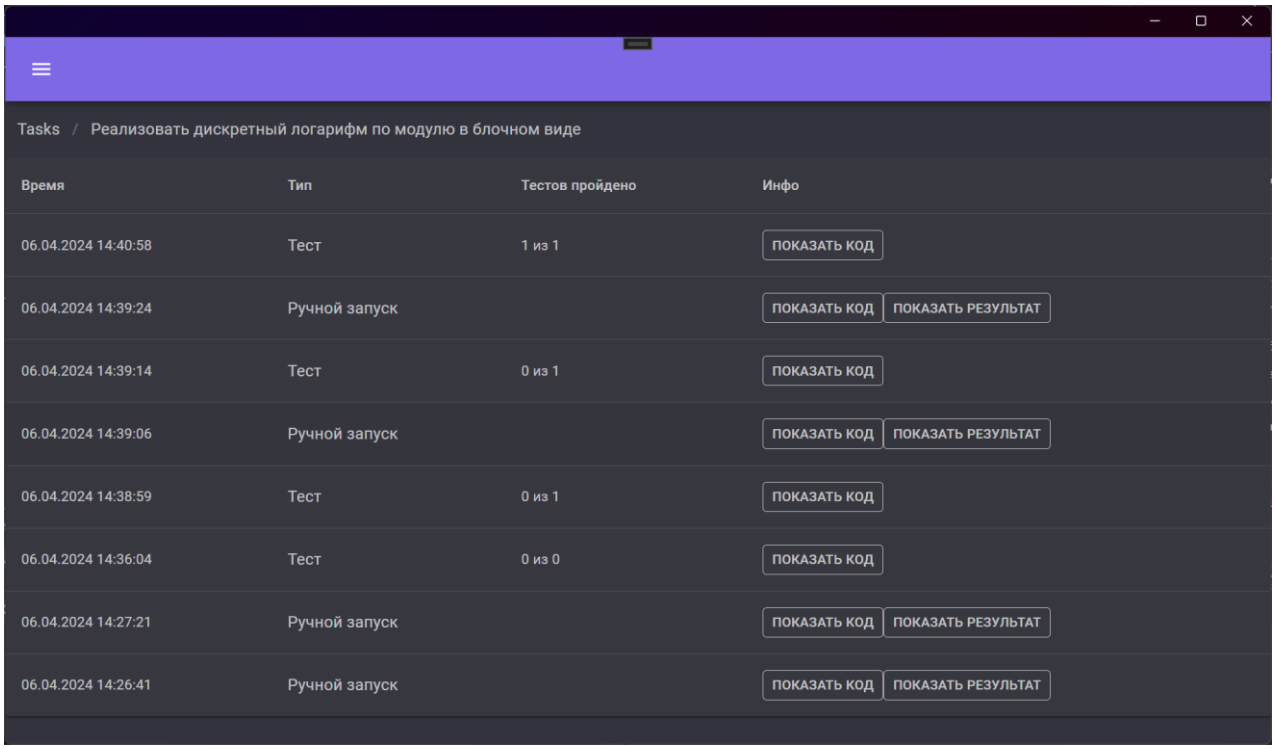


Также дополнительно присутствуют две кнопки: кнопка отображения истории запуска и кнопка отображения текущей программы виде скрипта Python.

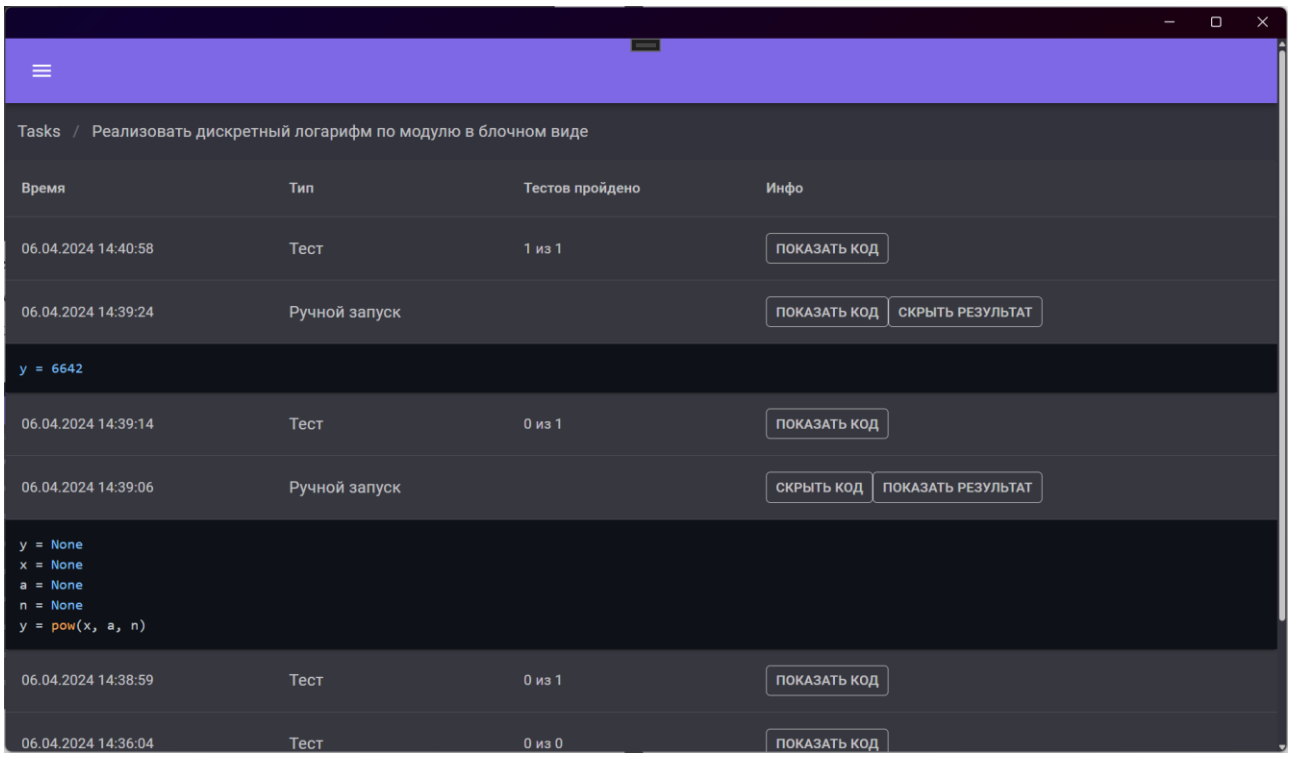
Кнопка отображения скрипта изображена как лист бумаги и расположена в правом верхнем углу экрана.



В истории запусков отражаются все попытки запуска программы. Отображается время, тип (ручной запуск или тест), удачно пройденные тесты и их общее количество, а также скрипт на момент запуска и результат для ручного запуска теста.



Путем нажатия соответствующих кнопок можно раскрыть информацию о результате выполнения или кода.

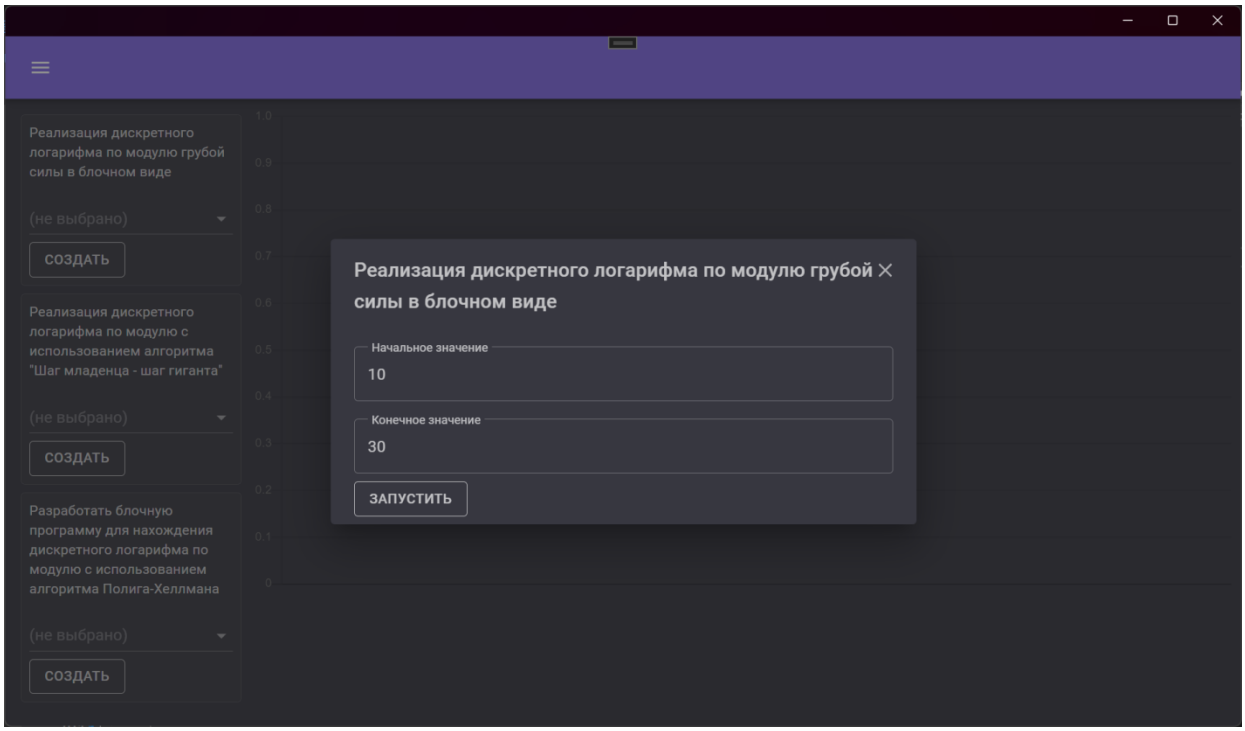


Окно тестирования эффективности алгоритма

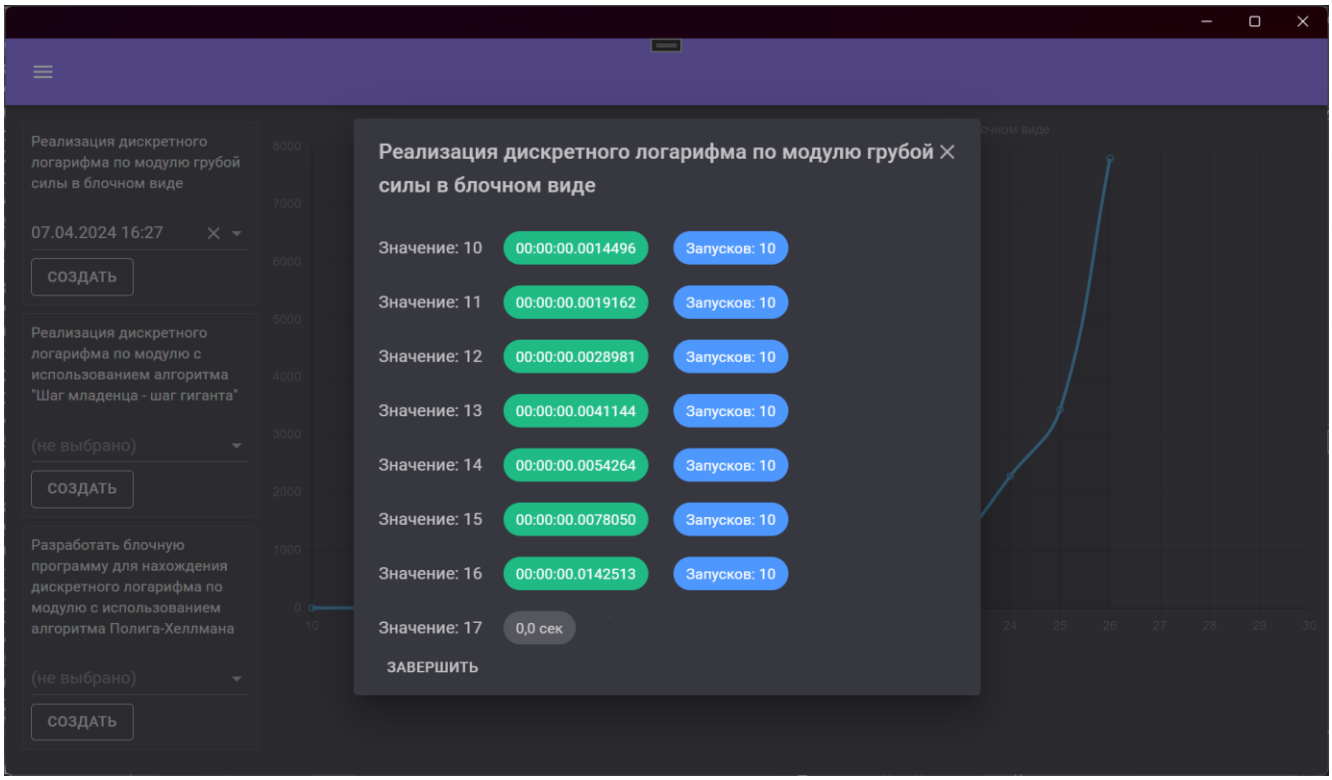
Окно тестирования эффективности работы алгоритма состоит из 2-х частей: слева располагается список задач, для которых можно запустить тест; справа – график, позволяющий выводить несколько тестов эффективности и производить сравнение.

Так как тесты эффективности могут занимать достаточно большое количество времени, их запуск производится вручную путем нажатия кнопки "Создать". Для измерения используется последний удачный запуск теста из вкладки "Задачи". Это позволяет оценить не только эффективность алгоритма, но и эффективности реализации.

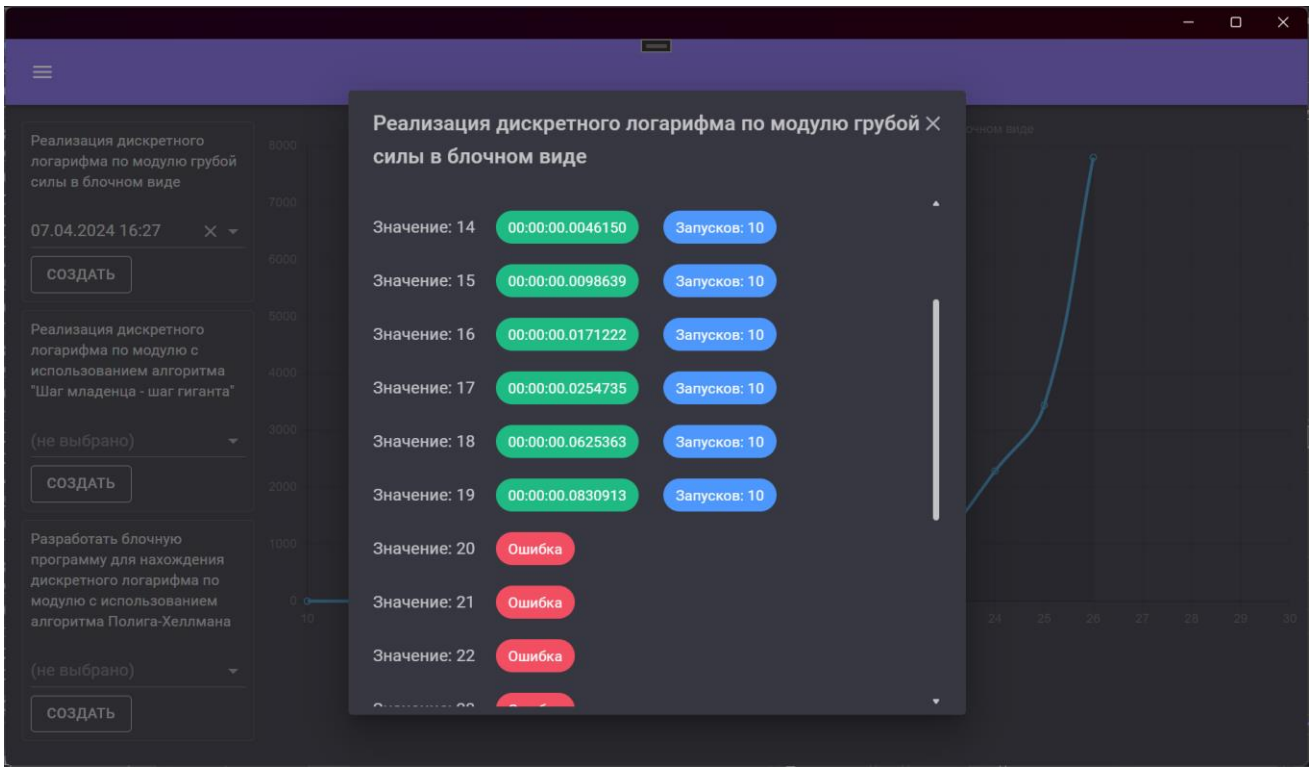
После нажатия кнопки "Создать" отобразится окно с выбором нужного диапазона значений оси *x*.



Открывается окно, отображающее в реальном времени статус тестирования. Текущая задача отмечена кольцом загрузки и таймером, показывающим затраченное время вместе с подготовительным скриптом. Рядом с итоговым временем выполнения находится информация о количестве запусков программы. Если задача завершилась с ошибкой, будет отображаться соответствующее сообщение напротив значения.



Также есть возможность принудительно завершить тестирование путем нажатия кнопки "Завершить". В данном случае текущий и последующие тесты будут со статусом "Ошибка". Запуск теста в любом случае сохранится и его можно будет посмотреть на графике. Значение, которые завершились ошибкой, будет отображаться на графике как разрывы. Это сделано с целью сравнения графиков, кратно отличающихся по эффективности.


Реализация дискретного логарифма по модулю грубой силы в блочном виде
07.04.2024 16:27
СОЗДАТЬ
Реализация дискретного логарифма по модулю с использованием алгоритма "Шаг младенца - шаг гиганта"
(не выбрано)
СОЗДАТЬ
Разработать блочную программу для нахождения дискретного логарифма по модулю с использованием алгоритма Полига-Хеллмана
(не выбрано)
СОЗДАТЬ
Реализация дискретного логарифма по модулю грубой силы в блочном виде
Значение: 14 00:00:00.0046150 Запусков: 10
Значение: 15 00:00:00.0098639 Запусков: 10
Значение: 16 00:00:00.0171222 Запусков: 10
Значение: 17 00:00:00.0254735 Запусков: 10
Значение: 18 00:00:00.0625363 Запусков: 10
Значение: 19 00:00:00.0830913 Запусков: 10
Значение: 20 Ошибка
Значение: 21 Ошибка
Значение: 22 Ошибка

Задания по лабораторной работе

***Задание* 1.** Разработать блочную программу, которая вычисляет степень числа по модулю заданного основания. Требуется решить:

$$b = a \wedge x \bmod p.$$

Используйте для этого следующие блоки: блок присвоения значения переменной, блок «остаток от _÷_», математическая функция возведения в степень («_^_»), блоки переменных.

***Задание* 2.** Разработать блочную программу проверки числа на его простоту. Для решения задания методом Рабина-Миллера используйте следующие блоки: присвоение значения переменной, «остаток от _÷_», математическая функция возведения в степень («_^_»), блоки переменных, блок объявления функции, блок «если», блок проверки четности числа, блок константного значения, блоки цикла, блок округления числа, блок деления («_÷_»), блок случайного целого числа в диапазоне, блок выхода из цикла или перехода к следующему значению, логические блоки «ИЛИ» и равенства значений, блок вызова функции, блоки констант «ИСТИНА» и «ЛОЖЬ».

***Задание* 3.** Реализация шифрования и дешифрования Эль-Гамаля с использованием ключей. Требуется вычислить $a = g \wedge k \bmod n$ и $b = m * (y \wedge k) \bmod n$, а также для дешифрования $m = b * a \wedge (n - 1 - p) \bmod n$.

Используйте следующие блоки: присвоение значения переменной, «остаток от _÷_», математическая функция возведения в степень («_^_»), блоки переменных, блок деления («_÷_»), блок умножения («_*_»), блок вычитания («_–_»), блок возведения в степень по модулю («_^_%_»).

***Задание* 4.** Реализация дискретного логарифма по модулю методом грубой силы в блочном виде. В данном задании предстоит найти x в диапазоне от 1 до $n - 1$, где $b = a \wedge x \bmod n$.

Используйте для этого следующие блоки: присвоение значения переменной, «остаток от _÷_», математическая функция возведения в степень («_^_»), блоки переменных, блок цикла, блок «если».

Задание 5. Реализация дискретного логарифма по модулю с использованием алгоритма «Шаг младенца - шаг гиганта».

Используйте следующие блоки: присвоение значения переменной, «остаток от _ ÷ _», математическая функция возведения в степень («_ ^ _»), блоки переменных, блок деления («_ ÷ _»), блок умножения («_ * _»), блок вычитания («_ - _»), блок возведения в степень по модулю («_ ^ _ % _»), блок целого квадратного корня, блок создания пустого словаря, блок установки значения в словаре по ключу, блок получения значения из словаря по ключу.

Совет: вместо списка используйте словарь, так как эта структура данных использует хэш таблицы в своей реализации. Так, после формирования рядов вам потребуется найти совпадающие значения во вложенном цикле, вместо формирования ряда шагов гиганта, во 2-м цикле будем сразу искать значения в словаре шагов младенца. Ключ – a ^ (i * m) mod n, а значение – индекс.

***Задание* 6(*).** Разработать блочную программу для нахождения дискретного логарифма по модулю с использованием алгоритма Полига-Хеллмана.

Используйте следующие блоки: присвоение значения переменной, «остаток от _ ÷ _», математическая функция возведения в степень («_ ^ _»), блоки переменных, блок деления («_ ÷ _»), блок умножения («_ * _»), блок вычитания («_ – _»), блок возведения в степень по модулю («_ ^ _ % _»), блок целого квадратного корня, блок создания пустого словаря, блок установки значения в словаре по ключу, блок получения значения из словаря по ключу, блоки получения списка ключей и значений из словаря, блок округления, блок объявления функции, блок вызова функции, блок проверки нечетности, логический блок сравнения значений, блок создания пустого списка, вставки значения в список и его получение.

***Задание* 7.** Проанализируйте эффективность алгоритмов дискретного логарифмирования по модулю на различных длинах ключей.

Контрольные вопросы

1. Что такое криптосистема с открытым ключом?
2. Каковы основные принципы работы алгоритма Эль-Гамаля?
3. Какие этапы включает в себя процесс шифрования и дешифрования в схеме Эль-Гамаля?
4. Что такое задача дискретного логарифмирования по модулю и какова её роль в криптографии?
5. Какие методы решения задачи дискретного логарифмирования по модулю вы знаете?
6. Можете ли вы описать метод “грубой силы” для решения задачи дискретного логарифмирования? Каковы его преимущества и недостатки?
7. Что такое метод “Шаги младенца – шаги гиганта” и как он используется для решения задачи дискретного логарифмирования?
8. Как алгоритм Полига-Хеллмана решает задачу дискретного логарифмирования?
9. Каковы основные различия между методом “грубой силы”, методом “Шаги младенца – шаги гиганта” и алгоритмом Полига-Хеллмана?