В.Н. Мирянова
А.Д. Мирянова

ПРАКТИКУМ В ENGEE. Основы программирования

Учебно-методическое пособие
для вузов

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ВЫСШАЯ ТЕХНОЛОГИЧЕСКАЯ ШКОЛА
«СЕВАСТОПОЛЬСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ИНСТИТУТ»

В.Н. Мирянова, А.Д. Мирянова

ПРАКТИКУМ В ENGEE.
Основы программирования

Учебно-методическое пособие
для вузов

Севастополь
СевГУ
2026

УДК 004.43:519.682(076)
ББК 32.97-018.1 я73
М64

Р е ц е н з е н т:
Д.В. Моисеев – д-р. техн. наук, доцент, заведующий кафедрой
«Информационные технологии и системы»,
декан факультета информационных технологий Высшей технологичной школы
«Севастопольский приборостроительный институт

Мирянова, В.Н.

М64 **Практикум в Engee. Основы программирования**: учебно-методическое пособие для вузов / В.Н. Мирянова, А.Д. Мирянова ; Севастопольский государственный университет. – Севастополь : СевГУ, 2026. – 77 с. – Текст : электронный. – ISBN 978-5-6055894-3-3

Учебно-методическое пособие посвящено отечественной платформе Engee, предназначенной для программирования, инженерных расчетов и симуляции различных процессов.

Цель пособия – оказание помощи студентам в подготовке и выполнении практических работ по дисциплинам IT-направленности. В пособии представлены теоретические, практические материалы и задания для самостоятельной работы студентов.

УДК 004.43:519.682(076)
ББК 32.97-018.1 я733

Учебно-методическое пособие рассмотрено и рекомендовано к изданию на заседании ученого совета Высшей технологической школы «Севастопольский приборостроительный институт» ФГАОУ ВО «Севастопольский государственный университет», протокол № 3 от 18 декабря 2025 г.

ISBN 978-5-6055894-3-3

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Engee представляет собой интегрированную платформу для проектирования сложных технических систем, которая сочетает в себе два ключевых компонента:

1. **Инструментарий для вычислений и программирования,** аналогичный Matlab, предоставляет готовую среду для анализа данных, инженерных расчётов и создания прикладного ПО.

2. **Среда визуального моделирования**, по принципу Simulink, позволяет конструировать математические модели систем управления и физических процессов с помощью блок-схем, с последующим полунатурным моделированием и генерацией кода для встраиваемых платформ.

Оба модуля тесно интегрированы: данные свободно передаются между редактором кода и моделями. Платформа обладает развитыми средствами автоматизации на встроенном инженерном языке, что упрощает управление симуляциями и анализ результатов. Важно отметить, что эта синергия реализована на уровне архитектуры, обеспечивая безопасность и целостность. Ранее столь комплексный подход среди CAE-систем[1] предлагался только в Matlab, теперь он доступен и в Engee.

Областями применения Engee могут быть:

- математические расчёты и визуализация;
- разработка и моделирование алгоритмов;
- анализ данных и научные исследования;
- создание сложных системных проектов.

Платформа охватывает широкий спектр задач: от машинного обучения и искусственного интеллекта до прикладных инженерных дисциплин, таких как автоматическое управление, обработка сигналов, радиолокация и связь. Решения на базе Engee востребованы в аэрокосмической, оборонной, автомобильной, финансовой отраслях, на транспорте и в образовании.

Ключевое архитектурное преимущество — это **клиент-серверная технология**. Все ресурсоёмкие вычисления выполняются на выделенном сервере (в облаке или локальной сети предприятия), а взаимодействие со средой происходит через браузер. Это кардинально упрощает внедрение и администрирование: ИТ-специалисты централизованно управляют вычислительными мощностями, а пользователи получают доступ к системе с

[1] **CAE** (*Computer-aided engineering*) — общее название для программ и программных пакетов, предназначенных для решения различных инженерных задач: расчётов, анализа и симуляции физических процессов.

любого устройства без установки клиентского ПО, что выгодно отличает Engee от Matlab.

Как продукт, созданный в России, Engee учитывает потребности отечественных специалистов. Интерфейс и вся документация, включая авторский перевод справки по языку программирования, полностью русифицированы. Платформа дополнена сотнями практических примеров, десятками бесплатных онлайн-курсов и сопровождается квалифицированной технической поддержкой [2].

ВВЕДЕНИЕ

Настоящее учебно-методическое пособие разработано с целью оказания комплексной помощи студентам преимущественно инженерно-технических и IT-специальностей в освоении фундаментальных и прикладных аспектов информационных технологий. В качестве основного инструментария в процессе обучения используется мощная российская платформа для математических вычислений и динамического моделирования — **Engee**. Пособие призвано объединить теоретические знания с их практическим применением, сформировав у учащихся навыки решения базовых задач программирования в современной программной среде.

Практическая составляющая пособия, являющаяся его ядром, включает в себя три последовательные лабораторные работы, которые знакомят студентов с базовыми возможностями платформы Engee. Каждая работа выстроена по единому, логически обоснованному принципу и содержит:

- теоретические сведения, в которых излагаются необходимые для выполнения работы концепции и принципы;
- четкие указания по выполнению заданий, позволяющие освоить интерфейс и основные функции системы;
- набор индивидуальных заданий для самостоятельного решения, обеспечивающих объективную проверку усвоенного материала;
- исчерпывающие требования к содержанию и оформлению отчета; приучающие студентов к соблюдению стандартов технической документации.
- перечень контрольных вопросов для самопроверки и закрепления пройденных тем.

В рамках первой работы студенты получают первичное представление о рабочем пространстве Engee, осваивают навигацию по разделам, изучают панели инструментов и основные элементы управления. Ключевой задачей является выполнение простейших арифметических и логических вычислений, что позволяет освоить синтаксис встроенного языка и получить первые навыки взаимодействия с системой.

Вторая работа посвящена одной из основных структур данных, используемых в инженерных и математических расчетах, — матрицам. Студенты изучают различные способы задания матриц (ручной ввод, генерацию с помощью функций, импорт из файлов), а также осваивают выполнение базовых матричных операций: сложение, умножение, транспонирование, вычисление определителя и обратной матрицы.

Третья работа знакомит учащихся с основами программирования, закладывая фундамент для автоматизации расчетов и создания сложных алгоритмов. В ходе работы рассматриваются такие конструкции, как операторы ветвления (if-else), циклы (for, while), написание пользовательских функций и скриптов. Это позволяет перейти от единичных вычислений к созданию целостных программ для решения специфических задач.

Для обеспечения корректности выполнения заданий и удобства проверки, всем студентам рекомендуется сохранять результаты своей работы в тех форматах файлов, которые предлагаются платформой Engee по умолчанию. Все созданные файлы в обязательном порядке должны быть приложены к итоговому отчету по каждой практической работе, так как они служат прямым доказательством выполненной работы и позволяют преподавателю провести всесторонний анализ полученных результатов.

В отчетах по практическим работам следует приводить:

• цель работы;

• формулировку заданий;

• тексты разработанных программ или их копии с экрана;

• подробные выводы о проделанной работе.

Файлы отчетов и файлы с программным кодом следует загружать в соответствующие элементы электронного образовательного ресурса вуза.

Практическая работа № 1
ЭЛЕМЕНТАРНЫЕ ВЫЧИСЛЕНИЯ В ENGEE

1.1. Цель работы

Знакомство с интерфейсом Engee. Выполнение элементарных вычислений.

1.2. Теоретические сведения

1.2.1 Интерфейс

Для начала работы с Engee необходимо зайти на сайт по адресу https://start.engee.com. Система требует регистрации. Зарегистрированные пользователи могут заходить по адресу https://engee.com, где будет предложено ввести логин или зарегистрироваться (рис. 1.1).

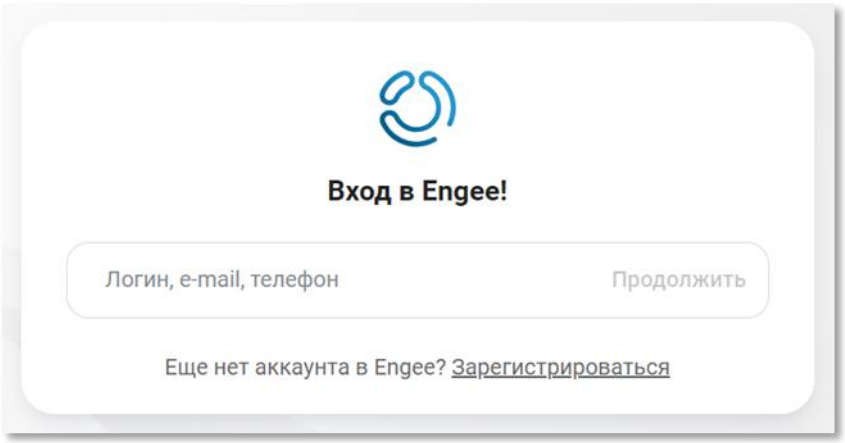


Рисунок 1.1 – Приглашение для входа в Engee

Панель управления, открывающаяся после входа в систему, представлена на рис. 1.2. На ней расположены следующие основные объекты:

- основное меню (данные профиля, настройки параметров входа, кнопка выхода и т.п.);
- кнопка запуска;
- кнопка перехода в сообщество (где представлены различные примеры использования Engee);
- кнопка перехода в документацию;
- окно уведомлений.

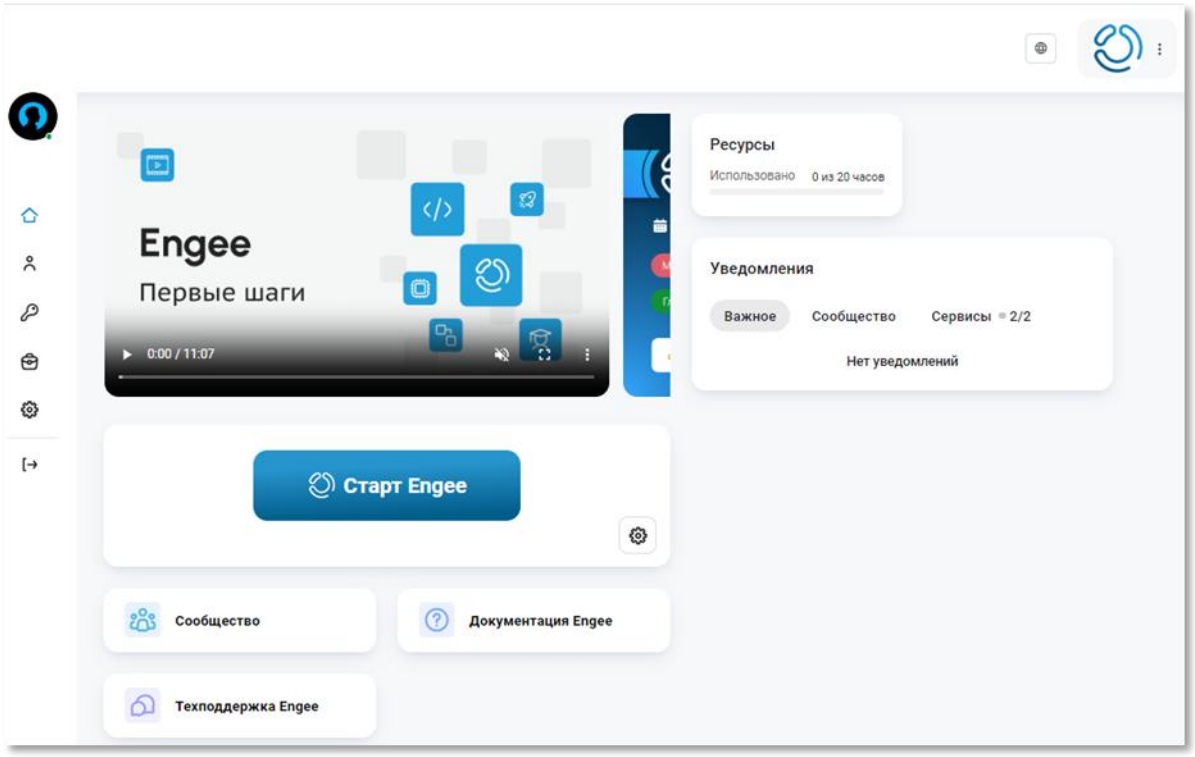


Рисунок 1.2 – Окно старта

После нажатия на кнопку «Старт Engee» открывается среда разработки (рис. 1.3).

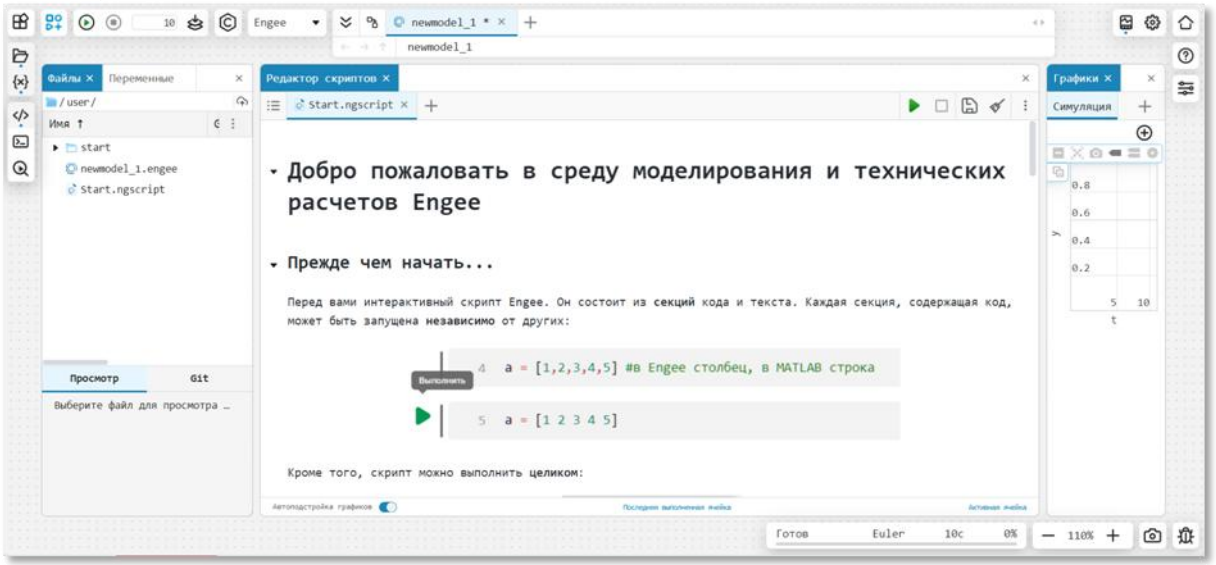


Рисунок 1.3 – Интерфейс системы

При первом входе сразу же будет доступна краткая инструкция к началу работы.

Интерфейс включает рабочее пространство (рис. 1.4) и разделы (кнопки) для вызова дополнительных окон и настроек.

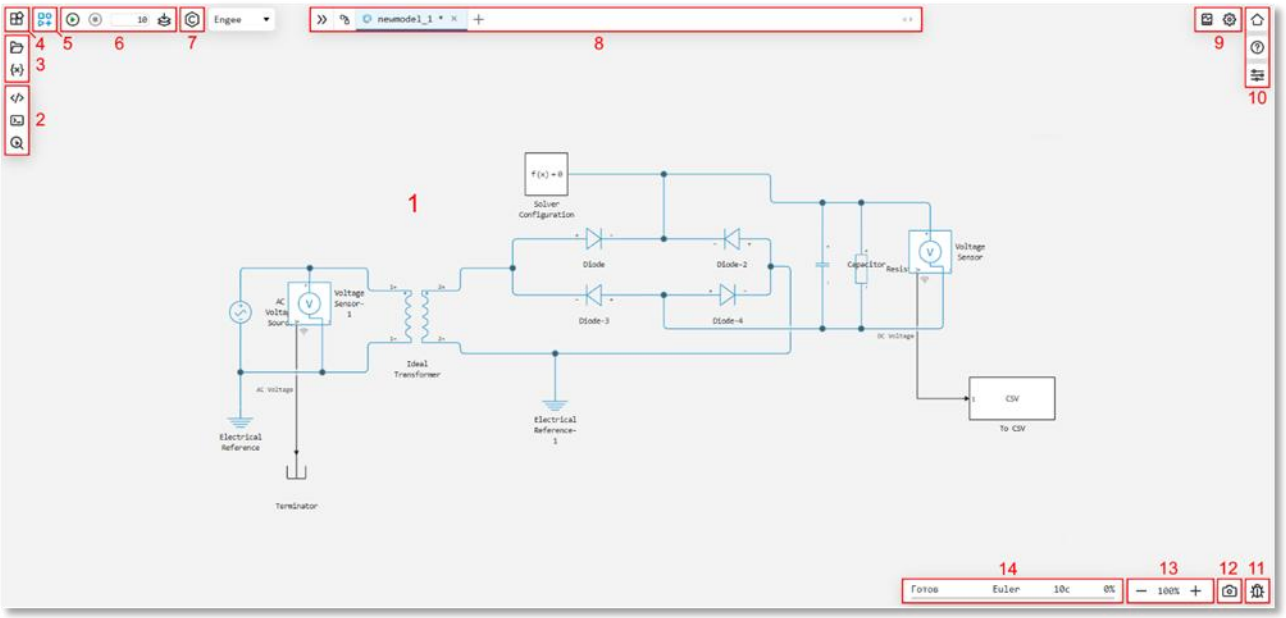


Рисунок 1.4 – Структура рабочего пространства [2]:

1. Рабочая область модели.
2. Раздел программирования.
3. Файлы и переменные.
4. Приложения.
5. Библиотека блоков.
6. Раздел управления симуляцией.
7. Генерация кода.
8. Панель навигации по моделям.
9. Настройки модели или блока, окно графиков.
10. Личный кабинет, ссылка на документацию и настройка интерфейса.
11. Кнопка **Обратная связь** (сообщить об ошибке).
12. Кнопка **Сделать скриншот.**
13. Настройка масштаба рабочей области модели.
14. Строка состояния.

Система позволяет настраивать интерфейс (расположение окон на рабочей области) или воспользоваться уже готовыми вариантами под разные задачи с помощью кнопки настройки интерфейса (рис. 1.4, блок 10):

- Среда моделирования — включает/отключает инструменты моделирования Engee.
- Язык — переключает интерфейс между русским и английским языками.
- Темная тема (экспериментально) — включает темную тему в рабочем пространстве.
- Сценарии — отображает предустановленный вариант интерфейса:
  1. Стандартный — открывает основные инструменты рабочего пространства Engee, а именно:
     - Файловый браузер с вкладками переменных и библиотеки блоков;
     - Редактор скриптов и командную строку;
     - Окна графиков и настроек.
  2. Программирование — открывает переменные и редактор скриптов.
  3. Тестирование — открывает редактор скриптов и окно графиков.
  4. Моделирование — открывает библиотеку блоков и настройки.
  5. Симуляция — закрывает все инструменты рабочего пространства.
- Сетка — включает сетку для рабочей области модели.
- Подписи всех блоков — включает подписи блоков модели.

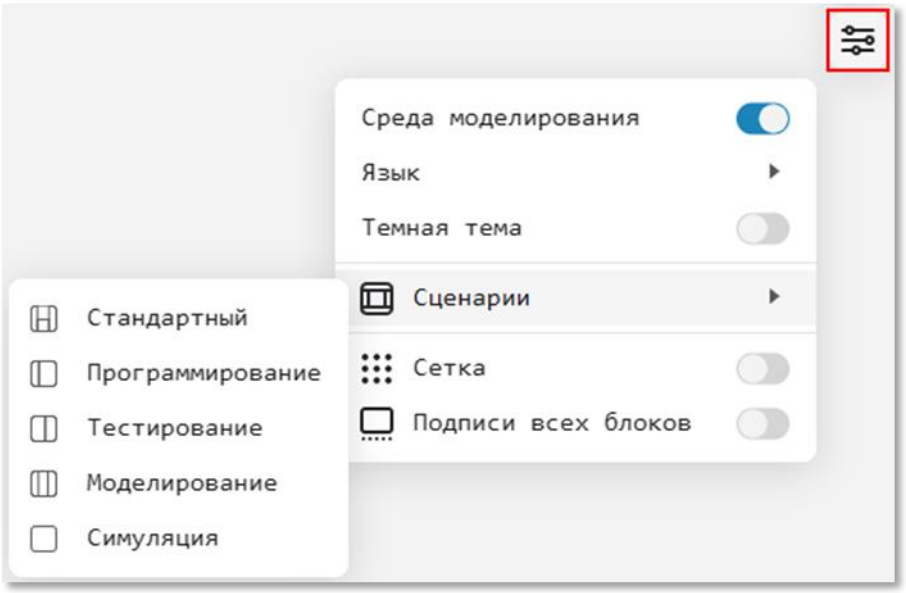


Рисунок 1.5 – Настройка интерфейса [2]:

С помощью кнопки в разделе программирования (рис. 1.4, блок 2) открывается **Командная строка**. Это инструмент для выполнения широкого спектра задач, от исполнения кода до создания сложных моделей в Engee. Командная строка интегрируется во все сценарии работы Engee.

Обычный режим командной строки позволяет интерактивно вводить команды, выполнять их и сразу же получать результат.

В рабочей области окна **Командной строки** находится строка ввода команд, отмеченная знаком

engee>,

после которого можно вводить числа, имена переменных и знаки операций, составляющие в совокупности выражение.

Окно **Командной строки** разделено на две принципиально различных зоны: зону просмотра и зону редактирования (рис. 1.6).

Исправление информации в *зоне просмотра* невозможно. Однако в ней можно выделить любой фрагмент текста, затем скопировать его в буфер обмена операционной системы.

Зона редактирования находится в строке командного окна, отмеченной знаком **engee>**. Все, что выше этого знака, изменению не подлежит.

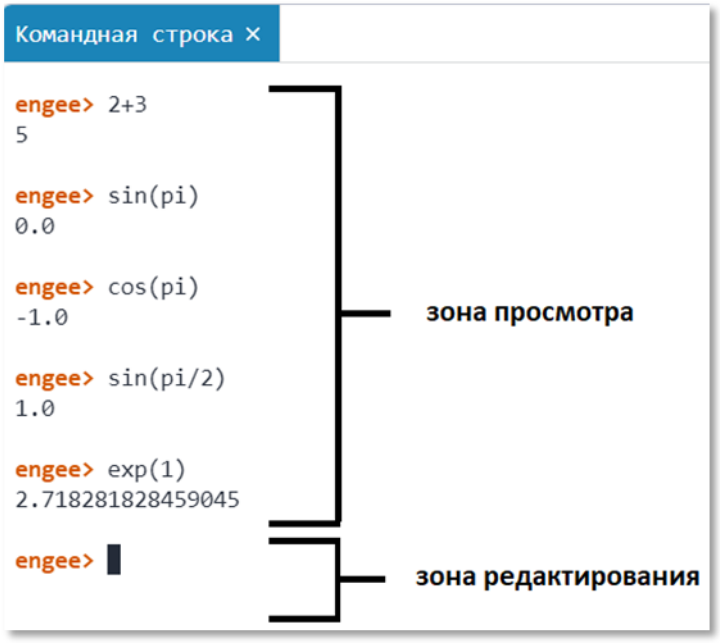


Рисунок 1.6 – Зонирование окна командной строки

В командной строке Engee доступны различные горячие клавиши, упрощающие навигацию и редактирование, например:

- ↑ — посмотреть предыдущую команду из истории команд,
- ↓ — посмотреть следующую команду из истории команд,
- ← — перемещает курсор на один символ влево,
- → — перемещает курсор на один символ вправо,
- Ctrl+← — переместить курсор на целое слово влево,
- Ctrl+→ — переместить курсор на целое слово вправо,
- Ctrl+A — переместить курсор в начало строки (аналогично для Alt+A),
- Ctrl+E — переместить курсор в конец строки, аналогично для Alt+E),
- Ctrl+K — удалить текст от курсора до конца строки,
- Ctrl+U — удалить текст от курсора до начала строки,
- Ctrl+C — прервать текущую команду,
- Ctrl+L — очистить экран терминала,
- Ctrl+Y — вставить последний удаленный текст,
- Ctrl+J — перевести строку на следующий абзац.

Очистка командного окна осуществляется командой ***Ctrl+L***, которая, однако, оставляет неизменным содержимое буфера команд и пространства переменных.

Для того чтобы узнать текущее значение любой переменной, достаточно набрать в **Командной строке** имя этой переменной и нажать клавишу ENTER, либо использовать **Окно переменных**, в котором отображаются все переменные, использованные в данном сеансе работы с пакетом.

Окно переменных {x} (рис. 1.4, блок 3) — это инструмент Engee, предназначенный для управления переменными. В нем содержится информация о переменных, созданных в процессе работы.

Переменная — это имя, ассоциированное (связанное) с некоторым значением. Значение — это число, строка, матрица или другой тип данных. Переменные в Engee доступны в командной строке, редакторе скриптов и окне **Переменные**. Переменные также могут быть переданы как параметры блоков среды моделирования.

Переменные можно создать через командную строку (рис. 1.7).

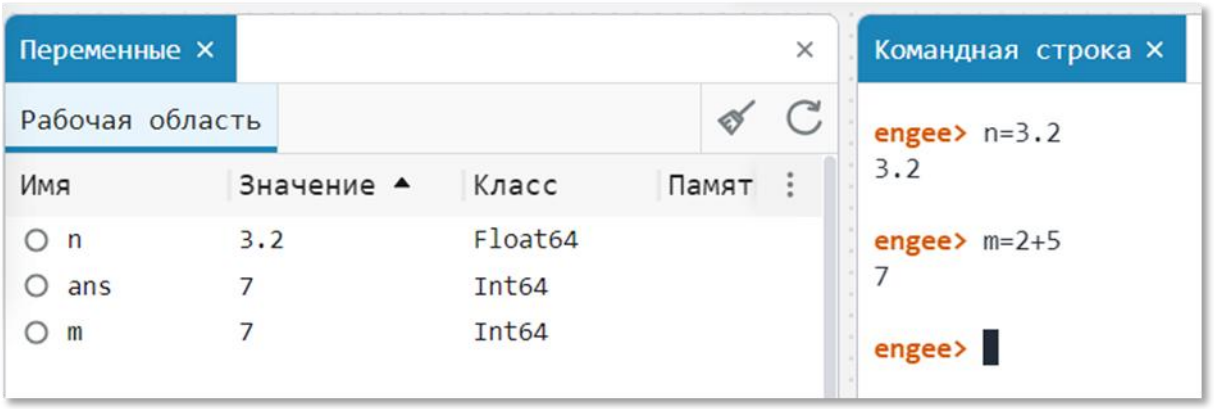


Рисунок 1.7 – Переменные, созданные в командной строке и отображенные в окне переменных

Следует отметить, что эффективность работы пакета будет снижается по мере увеличения объема рабочей области переменных, поэтому при отсутствии необходимости хранения в текущем сеансе работы некоторых переменных, их следует удалять из памяти. Это можно сделать, выделив нужную переменную в окне переменных и правой кнопкой мыши выбрав команду «Очистить» или нажав кнопку «Delete» на клавиатуре. Очистить все переменные можно, выделив их все, как в любом текстовом редакторе, и нажав кнопку «Delete» на клавиатуре, или с помощью кнопки .

После завершения сеанса работы с Engee все ранее вычисленные в командной строке переменные теряются. Для сохранения в файле содержимого рабочей области можно выделить все или необходимые переменные и, вызвав меню правой кнопкой мыши, выбрать команду «Сохранить как…».

Имена всех сохраняемых файлов должны быть написаны латинскими буквами и арабскими цифрами. Допускается использование знака подчёркивания. Рабочая область сохраняется в файлах с расширением **.mat** или **.jld2**.

1.2.1. Алфавит

В Engee используется язык программирования Julia, получивший признание благодаря своей универсальности. Он быстрый, у него удобный синтаксис и он отлично подходит для научных вычислений в различных областях.

В Julia, как и в других языках программирования высокого уровня, используются:

- прописные и строчные буквы латинского алфавита от A до Z, а также знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные символы (таблица 1.1).

Таблица 1.1 – Некоторые специальные символы языка Julia

Символ	Назначение
[ ]	Квадратные скобки используются для задания матриц и векторов, индексации массивов
,	Запятая или пробел используются для разделения элементов матриц и операторов в строке ввода
;	Точка с запятой в конце команды отменяет вывод результата на экран. Используется для записи команд в одной строке, разделения строк матриц
:	Двоеточие используется для указания диапазона изменения величины, а также для обозначения групповой операции над элементами матрицы
( )	Круглые скобки применяются для задания аргументов функций
.	Точка используется для отделения дробную части числа от целой, а также в составе операций, выполняемых поэлементно (.*, ./)
#	Знак означает начало однострочного комментария
#=	Комбинация знаков означает начало многострочного комментария
=#	Комбинация знаков означает конец многострочного комментария
' '	Апострофы используются для создания символьной переменной (n = 'x')
"	Двойные кавычки используются для ввода символьной строки (str = "That's string")

Подробнее можно узнать в документации Engee, в разделе «Пунктуация» [2].

Из символов алфавита формируются *лексемы языка* – минимальные значимые единицы текста в программе:

- идентификаторы;
- ключевые (зарезервированные) слова;
- знаки операций (+,-,* и т.д.);
- константы;
- разделители (скобки, точка, запятая, пробельные символы).

Границы лексем определяются другими лексемами, такими, как разделители или знаки операций, а также комментариями.

Знак операции – это один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются.

Основными объектами языка программирования являются:

- числа;
- переменные;
- выражения;
- операторы;
- константы;
- комментарии;
- функции.

Идентификатор (ID) – это имя программного объекта (константы, переменной, метки, типа, функции, модуля, поля в структуре).

В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания; первым символом ID может быть буква или знак подчеркивания, но не цифра, пробелы внутри ID не допускаются. Julia ***различает строчные и прописные буквы.***

Например, XX, Xx, xX, xx – имена четырёх различных переменных.

При именовании объектов следует придерживаться общепринятых соглашений:

ID переменной обычно пишется строчными буквами, например, index (для сравнения: Index – это ID типа или функции, а INDEX – константа);

Идентификатор должен нести какой-либо смысл, поясняя назначение объекта в программе, например, sum (сумма);

Если ID состоит из нескольких слов, как, например, birth_date, то принято либо разделять слова символом подчеркивания (birth_date), либо писать каждое следующее слово с большой буквы (birthDate);

Разделители идентификаторов объектов: пробелы; символы табуляции, перевода строки и страницы; комментарии (играют роль пробелов). Наличие разделителей не влияет на работу программы.

Ключевые (зарезервированные) слова не могут быть использованы в качестве идентификаторов.

Некоторые ключевые слова: baremodule, begin, break, catch, const, continue, do, else, elseif, end, export, false, finally, for, function, global, if, import, let, local, macro, module, quote, return, struct, true, try, using, while.

1.2.2. Числа, константы, переменные, выражения, операторы

Числа – основной базовый тип. Могут быть действительными или комплексными.

Ввод действительных чисел осуществляется по правилам, принятым в языках программирования высокого уровня, например, как

```
0  -1  845   1.35  0.00177   2.3466e10  -3567.2323e-16,
```

а комплексных чисел – путем записи строки следующего вида:

<значение_Re> + <значение_Im>im

где **значение_Re** – действительная часть комплексного числа, **значение_Im** – мнимая часть комплексного числа, im – системная константа, задающая мнимую единицу:

```
2.4e7 + 3.005im
```

В Julia имеется возможность создания упорядоченных числовых значений (ранжированных переменных), подчиняющихся закону арифметической прогрессии.

Они формируются по правилу:

<начальное значение>:<шаг приращения>:<конечное значение>

Эта запись порождает последовательность чисел, изменяющуюся от начального до конечного значения с заданным шагом. Если шаг не задан, то он принимает значение 1.

Константа – именованная область памяти, в которой хранятся данные определенного типа, эти данные неизменны в процессе работы программы. У константы есть имя и значение. Имя служит для обращения к области памяти, в которой хранится значение константы. Тип константы определяется по ее значению.

Константы – это предварительно определенные числовые или символьные значения, представленные уникальным именем. К числовым константам относятся числа, например, 12, –3.5, 52.37e–10, а к символьным – любые последовательности символов заключенные в двойные апострофы, например: “Summa”, “RES”.

Некоторые встроенные константы:

`pi` – число π;

`Inf` – бесконечность;

`-Inf` – минус бесконечность;

`NaN` (Not a Number) – нечисловое значение.

Подробнее можно узнать в документации Engee, в разделе «Целые числа и числа с плавающей запятой» [2].

Переменная – именованная область памяти, в которой хранятся данные определенного типа, которые можно изменить в любой момент работы программы.

У переменной есть *имя (идентификатор)* и *значение*. Имя служит для обращения к области памяти, в которой хранится значение.

Оператор – это выражение, которое обозначает выполнение определённой операции (действия) над операндами (данными). В качестве операндов могут использоваться переменные, функции и числовых константы.

С помощью *оператора присваивания*, вводимого знаком равно (=), переменным можно задавать определенные значения:

<имя переменной> = <выражение>[;]

Имена переменных должны начинаться с буквы и состоять из букв, цифр и знаков препинания.

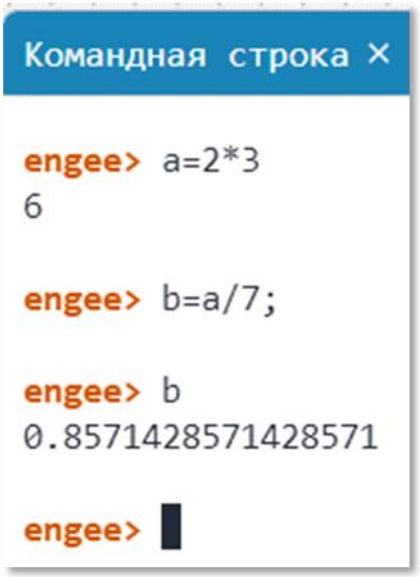


Рисунок 1.8 - Арифметические операции

В правой части оператора присваивания может стоять выражение, например: `y=(2-x)*(x+6).`

Если выражение встречается вне оператора присваивания, то его значение вычисляется и помещается в системную переменную `ans` (от answer).

Переменную `ans` можно использовать для задания новых выражений: `y = ans /5.`

Если оператор присваивания завершить символом «;» (точка с запятой), то результат на экране не дублируется; в противном случае – выводится на экран.

Типы переменных, как правило, заранее не объявляются. Они определяются выражением, значение которого присваивается переменной.

Арифметические выражения представляют собой последовательности чисел, констант, переменных и функций, объединенных знаками арифметических операций (рис. 1.8). Основные арифметические операторы приведены в таблице 1.2.

Таблица 1.2 – Арифметические операторы

Выражение	**Название**	**Описание**
+x	Унарный плюс	Операция тождественности
-x	Унарный минус	Отображает значения в их аддитивные инверсии.
x + y	Двоичный плюс	Выполняет сложение
x - y	Двоичный минус	Выполняет вычитание
x * y	Умножение	Выполняет умножение
x / y	Деление	Выполняет деление
x ÷ У	Целочисленное деление	x / y с округлением до целого числа
x \ y	Деление с инверсией	Эквивалентно y / x
x ^ y	Возведение в степень	Возводит x в степень y
x % y	Остаток	Эквивалентно rem (x.y)

Приоритет возведения в степень выше приоритетов умножения и деления, приоритет умножения и деления выше сложения и вычитания.

Для повышения приоритета операций используют круглые скобки. Операции одинакового приоритета выполняются в порядке слева направо, но круглые скобки при необходимости могут изменить этот порядок.

В Julia непосредственно перед переменной можно указать *числовой литерал*, подразумевающий умножение, например,

```
engee> x=3
3

engee> 2x^2 - 3x + 1
10
```

Приоритет числовых литеральных коэффициентов немного ниже, чем у унарных операторов, например, у оператора отрицания. Таким образом, $-2x$ анализируется как $(-2) * x$, а $\sqrt{2}x$ анализируется как $(\sqrt{2}) * x$. Однако в комбинации с экспонентой числовые литеральные коэффициенты анализируются аналогично унарным операторам. Например, $2\text{^}3x$ анализируется как $2\text{^}(3x)$, а $2x\text{^}3$ анализируется как $2 * (x\text{^}3)$.

Числовые литералы также работают как коэффициенты в выражениях с круглыми скобками:

```
engee > 2(x-1)^2 - 3(x-1) + 1
3
```

Приоритет числовых литеральных коэффициентов, используемых для неявного умножения, выше, чем у других двоичных операторов, таких как операторы умножения (*) и деления (/, \ и //). Это означает, например, что `1 / 2im` равно `-0.5im`, а `6 // 2(2 + 1)` равно `1 // 1`.

Кроме того, выражения в скобках могут использоваться в качестве коэффициентов для переменных, что подразумевает умножение выражения на переменную:

```
engee > (x-1)x
6
```

Однако для обозначения умножения нельзя использовать ни сопоставление двух выражений со скобками, ни размещение переменной перед выражением со скобками:

```
engee > (x-1)(x+1)
ERROR: MethodError: objects of type Int64 are not callable

engee > x(x+1)
ERROR: MethodError: objects of type Int64 are not callable
```

1.2.3. Некоторые встроенные элементарные функции

Функция – это объект с именем, выполняющий определенные преобразования над своими аргументами и возвращающий результаты этих преобразований. Для обращения к функции используется форма:

<имя результата> = <имя функции>(<список аргументов или их значений>).

Например, `y= cos(pi/2).`

Со списком элементарных функций можно познакомиться подробнее в документации Engee, в разделе «Математические операции и элементарные функции» [2].

Все элементарные функции должны записываться строчными (малыми) буквами.

Таблица 1.3 – Основные тригонометрические функции

Синтаксис	**Назначение**
sin(x)	синус числа x
asin(x)	арксинус (в радианах, в диапазоне от -π /2 к +π /2)
cos(x)	косинус
acos(x)	арккосинус (в диапазоне от 0 к π);
tan(x)	тангенс
atan(x)	арктангенс (в диапазоне от -π /2 к +π /2)
cot(x)	котангенс
acot(x)	арккотангенс
rad2deg(x)	преобразование угла x из радиан в градусы
deg2rad(x)	преобразование угла x из градусов в радианы

При работе с тригонометрическими функциями угол измеряется в радианах. Если необходимо вычислить, например, синус $\sin(30^o)$, то можно применить формулу перевода градусов в радианы:

```
engee> sin(30*pi/180)
0.49999999999999994
```

или встроенную функцию (табл. 3):

```
engee> sin(deg2rad(30))
0.49999999999999994
```

Также существуют тригонометрические функции для работы непосредственно в градусах: sind(x), cosd(x), tand(x), cotd(x), asind(x), acosd(x), atand(x), acotd(x), secd(x):

```
engee> sind(30)
0.5
```

Таблица 1.4 – Общие математические функции

Синтаксис	**Назначение**
abs(x)	Вычисляет модуль (абсолютная величина) числа x
sqrt(x)	Вычисляет квадратный корень числа x
round (x)	Вычисляет округление числа x до ближайшего целого
floor(x)	Округляет число x до целого в меньшую сторону
ceil(x)	Округляет число x до целого в большую сторону
sign(x)	Вычисляет знак числа x, , возвращая -1, 0 или +1.
rem(x,y)	Вычисляет остаток от деления x/y. Например, rem(25,4) даст ответ 1.
exp(x)	Вычисляет e^x
log(x)	Вычисляет натуральный логарифм числа x $\ln x$
log2	Вычисляет двоичный логарифм числа x
log10	Вычисляет десятичный логарифм числа x
log(n,x)	Вычисляет логарифм числа x по основанию n
factorial(n)	Вычисляет факториал n! (произведение всех положительных целых чисел, меньше или равное n, где n - неотрицательное целое число)

Если логарифм не является натуральным, десятичным или по основанию 2, то его также можно представить в виде отношения, например,

$$log_7 9 = \frac{ln9}{ln7} = \frac{lg9}{lg7},$$

которое с помощью операторов записывается следующим образом:

`log(9)/log(7)` или `log10(9)/log10(7)`.

1.2.4. Сообщение об ошибках и их исправление

При наличии ошибок в выражениях или командах Engee не только не выдает решение, но и указывает на наличие ошибки в виде текстового сообщения (рис. 1.9). Из него можно понять сущность ошибки, но иногда комментарии бывают настолько общими, что трудно установить место и содержание ошибки.

При сообщении об ошибке вычисления прекращаются.

В случае неопределенности результатов вычисления появляется сообщение NaN, что означает неопределенность, например, деление вида 0/0 или ∞/∞.

Устранение ошибки наиболее целесообразно не путем набора нового правильного выражения, а редактированием ошибочного.

```
engee> x=3.2
3.2

engee> x+2*(x+3


ERROR: syntax: incomplete: premature end of input
Stacktrace:
 [1] top-level scope
   @ none:1
```

```
engee> sqrt(-1)
ERROR: DomainError with -1.0:
sqrt will only return a complex result if called with a complex argument. Try sqrt(Com
plex(x)).
Stacktrace:
 [1] throw_complex_domainerror(f::Symbol, x::Float64)
   @ Base.Math ./math.jl:33
 [2] sqrt
   @ ./math.jl:677 [inlined]
 [3] sqrt(x::Int64)
   @ Base.Math ./math.jl:1491
 [4] top-level scope
   @ REPL[11]:1
```

Рисунок 1.9 – Примеры сообщений об ошибке

1.3. Порядок выполнения работы

1.3.1. Выполнить тренировочное задание:

- рассчитать алгебраическое выражение, осуществив его ввод в командной строке Engee,

$$\frac{17\left(\sqrt{5}-1\right)}{\left[15^2-13^2\right]}+\frac{5^7\log_{10}(e^3)}{®\sqrt{121}}+\ln(e^4)+\sqrt{11};$$

- переменные рабочей области сохранить в файле.

1.3.2. Вычислить значение выражения по варианту [3], выданному преподавателем. Варианты указаны в таблице 1.6. В работе использовать командную строку Engee. Результаты работы сохранить в файле.

Таблица 1.6 – Варианты к заданию 1.3.2

№ варианта	Выражение	Значения аргументов		
		a	***b***	***x***
1.	$y=\frac{1+\sin^2\left(b^3+x^3\right)}{\sqrt[3]{b^3+x^3}}$	-	2.5	1.28
2.	$y=\frac{\sqrt[3]{ax+b}}{\lg^2 x}$	1.35	0.98	1.14
3.	$y=\frac{1+\lg^2\frac{x}{a}}{b-e^{\frac{x}{a}}}$	2.0	0.95	1.25
4.	$y=\sqrt[4]{\left\|x^2-2{,}5\right\|}+\sqrt[3]{\lg x^2}$	-	-	1.25
5.	$y=\frac{a^x-b^x}{\lg\frac{a}{b}}\sqrt[3]{ab}$	0.4	0.8	3.2
6.	$y=\frac{b^3+\sin^2 ax}{\arccos\left(xbx\right)+e^{-x/2}}$	1.2	0.48	0.7
7.	$y=\frac{\lg\left(x^2-1\right)}{\log_5\left(ax^2-b\right)}$	1.1	0.09	1.2
8.	$y=\frac{\arccos\left(x^2-b^2\right)}{\arcsin\left(x^2-a^2\right)}$	0.05	0.06	0.2
9.	$y=\arcsin\left(x^a\right)+\arccos\left(x^b\right)$	2.0	3.0	0.11
10.	$y=a^{x^2-1}-\lg\left(x^2-1\right)+\sqrt[3]{x^2-1}$	1.6	-	1.2
11.	$y=\frac{a\sqrt{x}-b\log_5 x}{\lg\left\|x-1\right\|}$	4.1	2.7	1.2

№ варианта	Выражение	Значения аргументов		
		a	***b***	***x***
12.	$y=\sqrt{\frac{\lvert a-bx\rvert}{\lg^3 x}}$	7.2	4.2	1.81
13.	$y=\left(\arcsin^2 x+\arccos^4 x\right)^3$	-	-	0.26
14.	$y=\frac{\ln_a\lvert b^2-x^2\rvert}{\sqrt[5]{\lvert x^2-a^2\rvert}}$	2.0	1.1	0.08
15.	$y=\frac{a+tg^2 bx}{b+ctg^2 ax}$	0.1	0.5	0.15
16.	$y=\frac{(a+bx)^{2,5}}{1+\lg(a+bx)}$	2.5	4.6	1.1
17.	$y=\frac{\lg^2(a+x)}{(a+x)^2}$	2.0	-	1.2
18.	$y=\frac{\sqrt[3]{(x-a)^2}+\sqrt[5]{\lvert x+b\rvert}}{\sqrt[9]{x^2-(a+b)^2}}$	0.8	0.4	1.23
19.	$y=\left(\sin^3 x+\cos^3 x\right)\ln x$	-	-	0.11
20.	$y=a^{x^2-1}-\lg\left(x^2-1\right)+\sqrt[3]{x^2-1}$	2.25	-	1.2
21.	$y=\frac{a\sqrt[3]{x}-b\log_5 x}{\lg^3(x-1)}$	4.1	2.7	1.5
22.	$y=\sqrt[5]{\frac{a+bx}{\lg^3 x}}$	7.2	1.3	1.56
23.	$y=\sqrt[7]{\arcsin^4 x+\arccos^6 x}$	-	-	0.22
24.	$y=\frac{\log_a\left(b^2-x^2\right)}{\sqrt[3]{\lvert x^2-a^2\rvert}}$	2.0	4.1	0.77
25.	$y=\frac{\sqrt[3]{a}+tg^{4.5}bx}{\sqrt[5]{b}+ctg^{2.7}ax}$	0.1	0.5	0.33
26.	$y=\frac{\sin(a+bx)^{3.5}}{1+\cos[\lg(a+bx)]}$	2.5	4.6	1.15
27.	$y=\frac{tg\left[\lg^3(a+x)\right]}{\sqrt[7]{(a+x)^2}}$	2.0	-	1.08
28.	$y=\frac{\sqrt[3]{x-a}+\sqrt[5]{x+b}}{\sqrt[7]{x}-\sqrt[9]{x^2-(a+b)^2}}$	0.8	0.4	1.42
29.	$y=\frac{a\sqrt{x}-b\log_5 x}{\lg\lvert x-1\rvert}$	3.1	0.7	0.2

№ варианта	Выражение	Значения аргументов		
		a	***b***	***x***
30.	$y = \frac{\lg\left(x^2 - 1\right)}{\log_5\left(ax^2 - b\right)}$	2.1	0.05	2.3

1.4. Содержание отчета

1. Название работы.
2. Цель работы.
3. Формулировка тренировочного задания и задания по варианту.
4. Копии с экрана выполненных заданий 1.3.1, 1.3.2.
5. Файлы с сохраненными данными.

1.5. Контрольные вопросы

1. Из каких элементов состоит главное окно Engee?
2. Какова функция Командной строки?
3. На какие зоны делится окно Командной строки? Опишите их назначение.
4. Каково назначение команды ***Ctrl+L***?
5. Каким образом можно очистить одну переменную?
6. Каковы требования к имени сохраняемого файла?
7. Различает ли командной строке Engee строчные и прописные буквы в написании идентификаторов?
8. Приведите формы представления чисел, которые используются в Engee.
9. Каким образом записывается комплексное число? Приведите пример.
10. Перечислите известные вам встроенные константы командной строке Engee.
11. Дайте определение переменной, оператора, функции?
12. Что собой представляет оператор присваивания? Каков его синтаксис?
13. Приведите основные тригонометрические функции.
14. Каким образом вычислить модуль числа?
15. Каков приоритет выполнения операций в Engee?

Практическая работа № 2
ОПЕРАЦИИ С МАТРИЦАМИ В ENGEE

2.1. Цель работы

Изучение формы представления данных в Engee. Приобретение навыков выполнения матричных операций.

2.2. Теоретические сведения

Engee — система, предназначенная для моделирования и осуществления сложных вычислений.

Массив — упорядоченная, пронумерованная совокупность однородных данных. У массива должно быть имя.

Под вектором понимается одномерный массив чисел, а под матрицей — двумерный массив.

В Engee базовым языком является язык Julia. В нем переменные могут быть разных типов, которые устанавливаются в процессе работы проргаммы (но могут быть назначены программистом). Отдельное заданное число является скаляром (размер (1,)), вектор из N элементов — это объект Vector ((N,)), а вектор-строка, вектор-столбец или матрица — это объект Matrix размерностью NxM.

Доступ к элементам массива осуществляется при помощи индекса. В Engee нумерация элементов массивов начинается с единицы. Это значит, что индексы должны быть больше или равны единице.

2.2.1. Одномерные массивы

Для создания одномерного массива используют:

- операцию конкатенации,
- вызов специальных функций,
- операцию задания диапазона с шагом следования элементов (задается двоеточием).

Операция конкатенации (соединение, склеивание) обозначается с помощью квадратных скобок [ ].

При использовании операции конкатенации объединяемые в одномерный массив элементы располагаются между открывающей и закрывающей квадратными скобками и отделяются друг от друга либо пробелом, либо запятой.

Например, следующее выражение, использующее операцию конкатенации:

```
engee> x = [1, 2, 3]
3-element Vector{Int64}:
 1
 2
 3
```

или

```
engee> x = [1 2 3]
1×3 Matrix{Int64}:
1  2  3
```

формирует переменную с именем x, являющуюся одномерным массивом, состоящим из трех элементов (вещественных чисел). Здесь под конкатенацией понимается «соединение» трех чисел в один объект — вектор.

В векторе-строке элементы записываются в квадратных скобках, разделённые пробелами или запятыми (рис. 2.1, а).

В векторе-столбце элементы записываются в квадратных скобках, но разделённые символом точка с запятой (;) (рис. 2.1, б).

```
engee> x = [1 2 3]
1×3 Matrix{Int64}:
 1  2  3
```

а)

```
engee> x = [1; 2; 3]
3-element Vector{Int64}:
 1
 2
 3
```

б)

Рисунок 2.1 – Задание вектора-строки (а) и вектора-столбца (б)

Операция индексации позволяет обратиться к тому или иному элементу массива чтобы поменять его значение на какое-то другое. Чтобы обратиться к отдельному элементу матрицы или массива, после имени массива необходимо указать в квадратных скобках индекс (номер) элемента. Третий элемент одномерного массива x обозначается как x[3], первый элемент — как x[l], второй элемент — как x[2]. Например, следующее выражение, использующее операцию индексации, позволяет поменять значения одномерного массива, состоящего из трех элементов (вещественных чисел):

```
engee> x[1] = 1;
engee> x[2] = 2;
engee> x[3] = 3;
engee> x
1×3 Matrix{Int64}:
1  2  3
```

Массивы можно заполнять в цикле, но этот метод создания массивов не является эффективным и проигрывает в быстродействии всем остальным операциям.

```
engee> @time z = [1 for i in 1:3]
 0.041841 seconds (34.75 k allocations: 2.237 MiB, 69.71%
compilation time)
3-element Vector{Int64}:
 1
 1
 1
```

Создание массива вызовом специальных функций существенно увеличит быстродействие работы Engee (табл. 2.1).

Например, для массива z можно перед присваиваниями сделать следующий вызов функции ones:

```
engee>@time z = ones( 3 )
0.000003 seconds (1 allocation: 80 bytes)
```

```
3-element Vector{Float64}:
 1.0
 1.0
 1.0
```

Или с помощью окна командной строки (рис. 2.2).

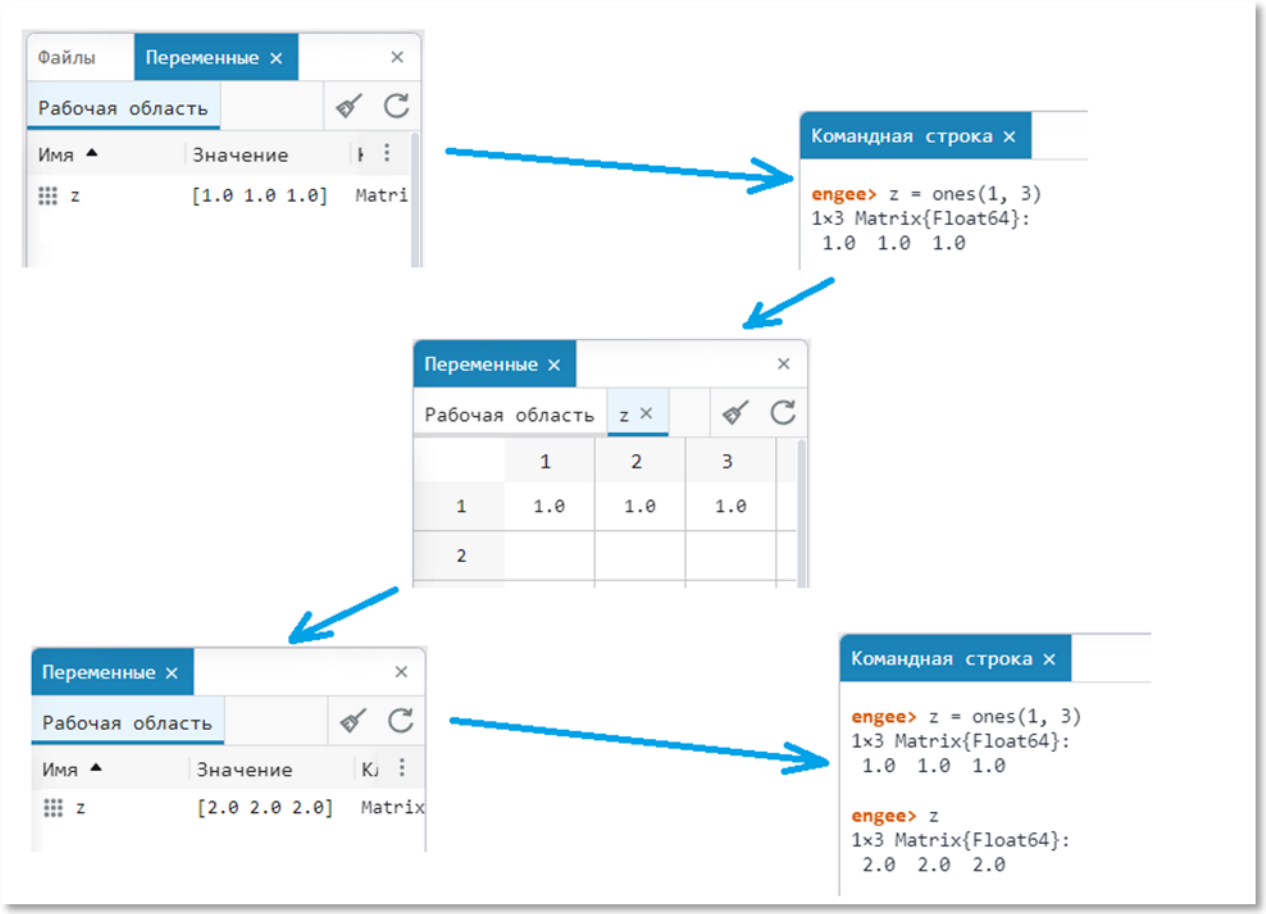


Рисунок 2.2 – Создание массива вызовом специальных функций и с помощью командной строки

Таблица 2.1 – Некоторые команды формирования матриц

Команда (синтаксис)	**Описание**
[A B]	Горизонтальная конкатенация (объединение) матриц A и B.
ones(m,n)	Создаёт матрицу M x N с единичными элементами.
zeros(m,n)	Создаёт матрицу M x N с нулевыми элементами
rand(m,n)	Создает матрицу размером M x N из случайных чисел, равномерно распределенных в диапазоне от 0 до 1.
Matrix(I, M, N)	Создаёт матрицу M x N с единичными элементами по главной диагонали и остальными нулевыми элементами (необходимо подключить библиотеку LinearAlgebra).
inv(A)	Вычисляет матрицу, обратную матрице A.
A'	Вычисляет транспонированную матрицу A^T.
eigvals(A)	Вычисляет массив собственных чисел матрицы A, т.е. корни характеристического полинома det(sI − A).

Операцией формирования диапазона числовых значений, например, можно создать одномерный массив чисел в диапазоне от 1 до 3 с приращением 1:

```
engee> z = 1:1:3
1:1:3

engee> collect( z )
3-element Vector{Int64}:
1
2
3
```

В формируемый массив сначала включается левая граница диапазона. Затем к этому числовому значению прибавляется приращение, которое указывается после первого двоеточия. Если сумма не превосходит верхней границы диапазона, то она включается в качестве элемента в формируемый массив. Это все повторяется до тех пор, пока очередное числовое значение не превысит верхнюю границу.

Эта операция также часто используется при построении графиков в ограниченном диапазоне изменения аргументов.

2.2.2. Двумерные массивы

Для создания двумерного массива (матрицы) в Engee используют (рис. 2.3):

- операцию конкатенации,
- вызов специальных функций.

Значения элементов матрицы вводятся в квадратных скобках [ ] по строкам. При этом элементы строки матрицы разделяются пробелом, а строки отделяются одна от другой при помощи точкой с запятой. Первым указывается число строк, вторым — число столбцов.

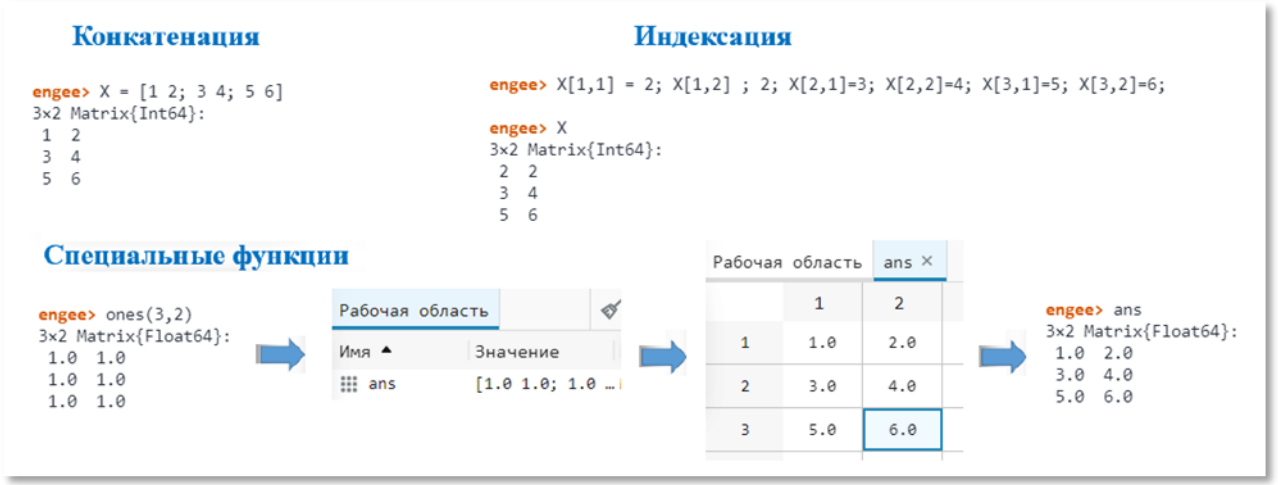


Рисунок 2.3 – Основные способы создания двумерного массива

2.2.3. Действия с матрицами

При работе с матрицами можно использовать два вида операторов:

- матричные – производят действия по правилам матричной алгебры;
- поэлементные – производят действия над соответствующими элементами матриц, при этом размеры матриц должны быть одинаковыми. От матричных операций отличаются точкой перед знаком операции.

Поэлементные операции могут также быть постолбцовыми и построчными. Такие операции допускают, чтобы размеры операндов имели разную размерность. Например:

```
engee> ones(3,1) .+ ones(3,3)
3×3 Matrix{Float64}:
2.0  2.0  2.0
2.0  2.0  2.0
2.0  2.0  2.0
```

Для удаления строк и столбцов требуется создать новую матрицу, из которой исключены удаляемые элементы. Например, удалим первую строчку следующей матрицы:

```
engee> A = [2 0 3; 1 1 4; 6 1 3]
3×3 Matrix{Int64}:
2  0  3
1  1  4
6  1  3

engee> A[Not(1), :]
2×3 Matrix{Int64}:
1  1  4
6  1  3
```

Или проще:

```
engee> A = [2 0 3; 1 1 4; 6 1 3];
engee> A = A[2:3, :];
2×3 Matrix{Int64}:
1  1  4
6  1  3
```

Аналогичным образом удаляются и столбцы:

```
engee>A = [2 0 3; 1 1 4; 6 1 3];
engee>A[:, 1] ;
3-element Vector{Int64}:
2
1
6
```

Самый простой способ удаления строк или столбцов – использование редактора в окне Переменные (рис. 2.4).

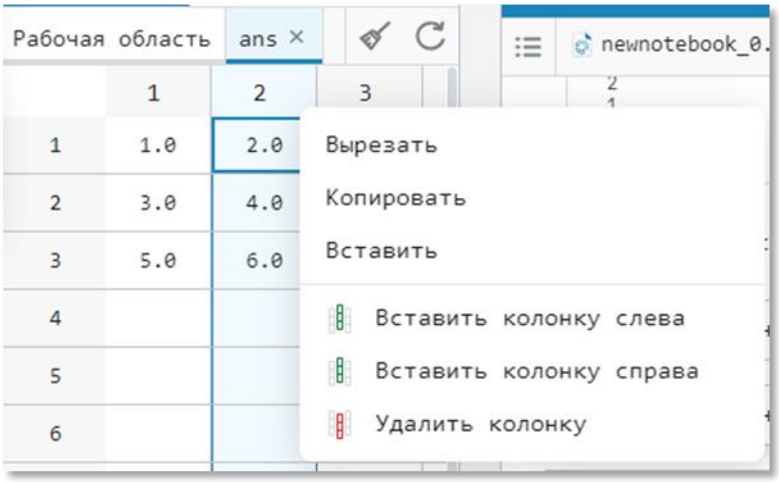


Рисунок 2.4 – Удаление элементов массива

При выполнении матричных операций необходимо помнить следующие правила:

- при сложении или вычитании матрицы должны иметь одинаковые размеры;
- при умножении матриц число столбцов первой матрицы должно совпадать с числом строк второй матрицы;
- при обращении матрица должна быть квадратной.
- приоритет операций: сначала выполняется операция транспонирования, затем возведения в степень, потом умножение и деление, а в последнюю очередь – сложение.

Невыполнение этих условий вызывает появление в командном окне сообщения об ошибке (рис. 2.5).

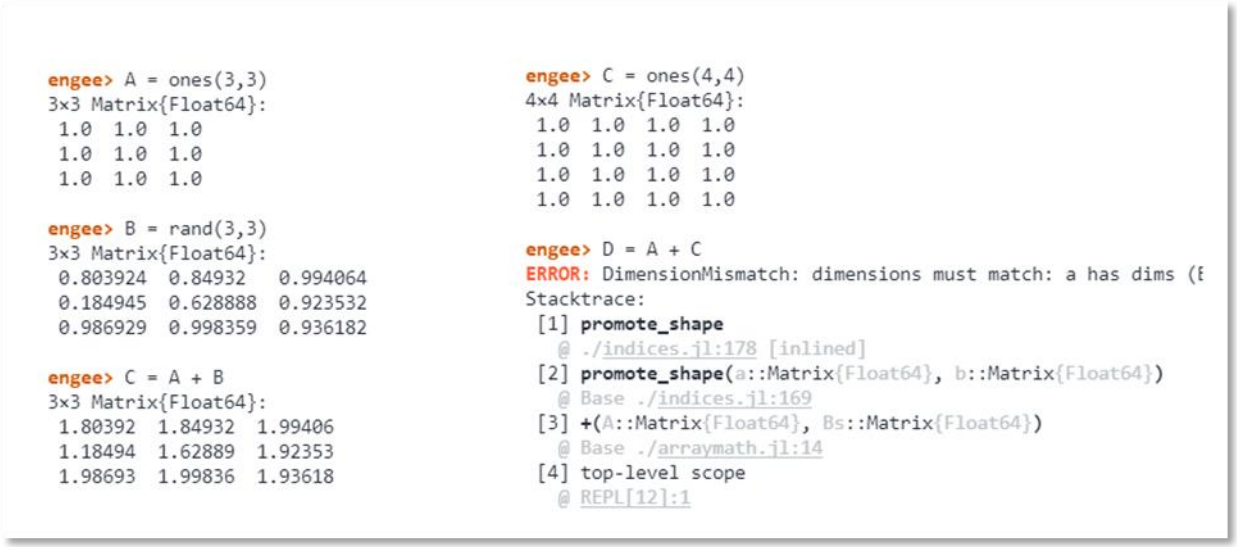


Рисунок 2.5 – Сложение матриц: с соблюдением размерностей, а);
без соблюдения размерностей, б)

Базовые операции над векторами и матрицами – сложение, вычитание, умножение матрицы на число, умножение матрицы на матрицу – выполняются

с применением обычных знаков арифметических операций, по правилам, предусмотренным в математике. Дополнительно используют матричные арифметические операции, представленные в таблице 2.2.

Таблица 2.2 – Основные команды поэлементных операций над матрицами

Команда (синтаксис)	**Описание**
+ (A+B)	Сложение
- (A-B)	Вычитание
* (A*B)	Матричное умножение
.* (A.*B)	Поэлементное умножение массивов
^ (A^a)	Возведение матрицы A в степень a
.^ (A.^a)	Поэлементное возведение массива в степень
/ (A / B)	Деление матриц слева направо
./ (A ./ B)	Поэлементное деление массивов слева направо (приводит к A(k,m)/B(k,m))
\ (A \ B)	Деление матриц справа налево
.\ (A .\B)	Поэлементное деление массивов справа налево (приводит к B(k,m)/A(k,m))

Кроме базовых существуют и другие операции над матрицами, некоторые из них приведены в таблице 2.3.

Таблица 2.3 – Основные команды поэлементных операций над матрицами

Команда (синтаксис)	**Описание**
[A B]	Горизонтальная конкатенация матриц A и B
A[m:n]	Возвращает элемент на пересечении строки m и столбца n матрицы A
A[:,n]	Возвращает все элементы из столбца n матрицы A
A[n,:]	Возвращает все элементы из строки n матрицы A
A[:,m:n]	Возвращает элементы из всех строк между столбцами m и n матрицы A
A[m:n,:]	Возвращает элементы из всех столбцов между строками m и n матрицы A
A[m:n,p:q]	Возвращает элементы из строк от m до n и столбцов от p до q матрицы A
A'	Вычисляет транспонированную матрицу A^T («э» в англ. раскладке)
ndims(A)	Вычисляет размерность массива (одномерный – 1, двумерный – 2)
size(A)	Вычисляет размер массива (количество строк и столбцов)
inv(A)	Вычисляет матрицу, обратную матрице A.
det(A)	Вычисляет определитель матрицы A
length(A)	Возвращает количество элементов в одномерном массиве A
sum(A)	Сумма элементов матрицы A
sum(A,dims=1) или sum(A,dims=2)	Сумма элементов по столбцам или по строкам матрицы A
prod(A)	Произведение элементов матрицы A (построчные операции аналогичны sum
maximum(A)	Нахождение максимального элемента каждого столбца матрицы A
minimum(A)	Нахождение минимального элемента каждого столбца матрицы A
sort(A, dims=1)	Сортировка элементов матрицы A по столбцам (сверху вниз по возрастанию)

Примеры матричных операций [2] приведены в приложении А, основные алгебраические операции над матрицами — в приложении Б.

2.3. Порядок выполнения работы

2.3.1. Выполнить тренировочные задания в командном окне Engee, повторив операции, представленные в пункте 2.2.

2.3.2. Выполнить операции с матрицами по варианту, выданному преподавателем. Варианты указаны в таблице 2.4 [3]. В работе использовать командное окно Engee.

Таблица 2.4 – Варианты к заданию 2.3.2

№ варианта	Задание
1.	$C = A + B^T \cdot B; \quad A = \begin{bmatrix} 100 & 100 \\ 200 & 200 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 0 & 1 \end{bmatrix}$
2.	$C = A + B \cdot B^T \quad A = \begin{bmatrix} 100 & 100 \\ 1 & 1 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \end{bmatrix}$
3.	$C = A^T \cdot B \quad A = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$
4.	$C = B \cdot A; \quad A = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$
5.	$C = A^T \cdot B^T; \quad A = \begin{bmatrix} 1 \\ 2 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 0 & 2 \end{bmatrix}$
6.	$C = B^T \cdot A; \quad A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 0 & 2 \end{bmatrix}$
7.	$C = A \cdot B \cdot B^T; \quad A = 2; \quad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 0 & 1 \end{bmatrix}$
8.	$C = B \cdot B^T \cdot A; \quad A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

№ варианта	Задание
9.	$C = A^T B^T B; \quad A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$
10.	$C = A + B \cdot B^T; \quad A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}; \quad B = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 0 & 2 \end{bmatrix}$
11.	Из произвольной матрицы A(5x5) выделить минор, который образуется в результате вычеркивания из этой матрицы четвертой строчки и первого столбца.
12.	Задать произвольную матрицу A(6x4). Требуется выделить из матрицы вторую строку по порядку.
13.	Задать произвольные матрицы A(4x4) и B(5x5). Требуется получить из этих матриц два вектора. Первый вектор должен совпадать с четвёртым столбцом матрицы A, а второй - с первым столбцом матрицы B.
14.	Из произвольной матрицы A(4x4) выделить минор, который образуется в результате вычеркивания из этой матрицы третьей строчки и второго столбца.
15.	Требуется сформировать диагональную квадратную матрицу C(5x5). Значения элементов главной диагонали должны совпадать с номером строки.
16.	Задать произвольную матрицу A(3x3). Требуется получить из этой матрицы два вектора. Первый вектор должен совпадать с первым столбцом матрицы A, второй – с третьим столбцом матрицы A.
17.	Задать произвольные матрицы A(2x2) и B(4x2). Требуется объединить эти матрицы в одну матрицу C(6x2), причем в новой матрице в качестве первых строк должны быть строки матрицы B, а за ними должны следовать строки матрицы A.
18.	Задать произвольные матрицы A(4x3) и B(4x2). Требуется объединить эти матрицы в одну матрицу C(4x5), причем первыми столбцами новой матрицы должны быть столбцы матрицы A, а справа от этих элементов следовать столбцы матрицы B.
19.	Сформировать произвольную матрицу C(6x6). Требуется найти сумму элементов в пятом столбце матрицы.
20.	Задать произвольные матрицы A(4x5) и B(4x2). Требуется выделить из матрицы A первый столбец по порядку и объединить его с матрицей B дописыванием справа.
21.	Задан вектор x=1:9. Получить из него матрицу 3-го порядка, в каждой строке которой записаны последовательно элементы вектора.
22.	Задан вектор x=1:4. Создать матрицу 4-го порядка, элементы каждой строки (столбца) матрицы являются элементами вектора.

№ варианта	Задание
23.	Задан вектор x=1:4. Создать матрицу 4-го порядка, на диагоналях которой стояли бы элементы вектора. Если диагональ короче size(x), то заполнение начинать с 1-го элемента вектора x.
24.	Задан вектор x=1:16. Получить из него матрицу 4-го порядка, в каждом столбце которого записаны последовательно элементы вектора.
25.	Задан вектор x=1:3. Создать матрицу 3-го порядка, на диагоналях которой стояли бы элементы вектора. Если диагональ короче size(x), то заполнение начинать с 1-го элемента вектора x.
26.	Задать произвольную матрицу A(7×5). Требуется выделить из матрицы подматрицу, образованную пересечением строк 2-4 и столбцов 1-3.
27.	Требуется сформировать антидиагональную квадратную матрицу D(4×4). Значения элементов побочной диагонали должны совпадать с квадратами натуральных чисел от 1 до 4.
28.	Задать произвольные матрицы A(3×4) и B(2×4). Требуется объединить эти матрицы в одну матрицу C(5×4), причем в новой матрице в качестве первых строк должны быть строки матрицы A, а за ними должны следовать строки матрицы B.
29.	Из произвольной матрицы A(6×6) выделить минор, который образуется в результате вычеркивания из этой матрицы первой и пятой строки, а также второго и четвертого столбца.
30.	Задать произвольную матрицу A(5×3). Требуется получить из этой матрицы два вектора: первый вектор должен содержать элементы последней строки матрицы A, второй вектор - элементы среднего столбца матрицы A.

2.4. Содержание отчета

1. Название работы.
2. Цель работы.
3. Формулировка тренировочного задания и задания по варианту.
4. Копии с экрана выполненных заданий 2.3.1, 2.3.2.
5. Файл с результатами работы.

2.5. Контрольные вопросы

1. В чем отличие матрицы от вектора?
2. Каким термином можно объединить понятия вектора и матрицы?
3. Сопоставьте понятия «одномерный», «двумерный» массив и «вектор», «матрица».
4. Каким образом в Engee можно задать матрицу?
5. В чем отличие матричных и поэлементных операций?
6. Какая матрица называется единичной и с помощью какого оператора можно её создать в Engee?
7. Что такое определитель матрицы и как он вычисляется в Engee.
8. С помощью каких операторов можно в Engee найти максимальный и минимальный элементы матрицы?

9. Каким образом задаются двумерные массивы?
10. Как использовать окно Переменные для работы с матрицами?

Практическая работа № 3
ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ В ENGEE

3.1. Цель работы

Изучение принципов программирования в Engee и выработка навыков написания простейших программ с использованием редактора скриптов.

3.2. Теоретические сведения

Основой любой CAE (Computer-aided engineering)-системы является язык программирования. Он должен быть прост, удобен и понятен для инженеров и разработан для технических вычислений и анализа данных. Все CAE-системы работают на языках, удовлетворяющих этим требованиям. Engee, как и MATLAB, тоже выстроен вокруг специализированного инженерного языка, который называется Julia.

Julia – достаточно молодой (появился в 2009 году), но динамично развивающийся язык, его синтаксис прост и похож на язык MATLAB. Он быстр, эффективен и является свободным программным обеспечением.

Его высокоуровневый синтаксис позволяет легко выражать сложные алгоритмы, что делает язык более доступным и придает ему значительные выразительные возможности.

Синтаксис Julia интуитивно понятен и прост в изучении. Переменные можно присваивать без объявления их типа, язык поддерживает все распространенные алгоритмические структуры (циклы, условия), все распространенные структуры данных (многомерные матрицы, словари), а для сложных типов данных доступно множество бесплатных библиотек.

Для написания и запуска кода в Engee используются **Командная строка** или **Редактор скриптов** — удобный инструмент для пошаговой работы с кодом, визуализациями и комментариями.

Среда разработки интерактивных скриптов Engee поддерживает несколько языков программирования: Julia, Python, MATLAB, C/C++ и др., но основным является Julia.

3.2.1. Работа в редакторе скриптов

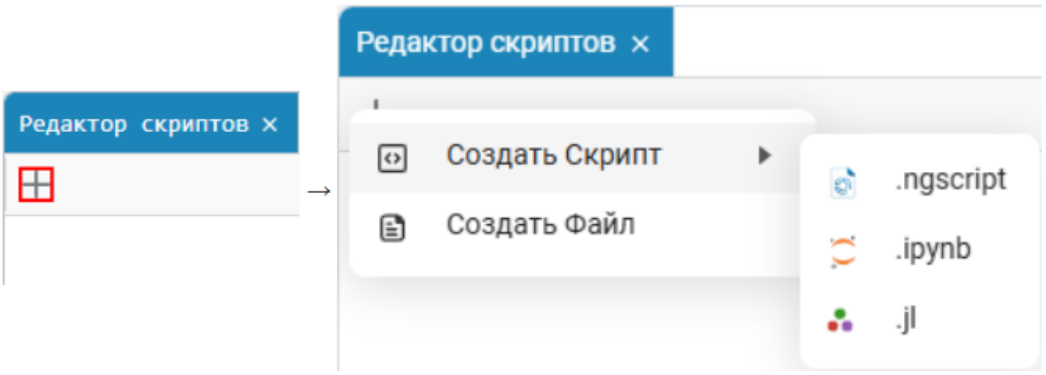
Интерактивные скрипты — это особый формат документов, в которых код, текст и графика объединены в единую рабочую структуру. Открыть редактор скриптов можно прямо из интерфейса Engee [2] — достаточно нажать на иконку </> в панели инструментов.

Чтобы создать скрипт, нажмите **+** в редакторе скриптов и выберите нужный формат, как показано на рис. 3.1, а.

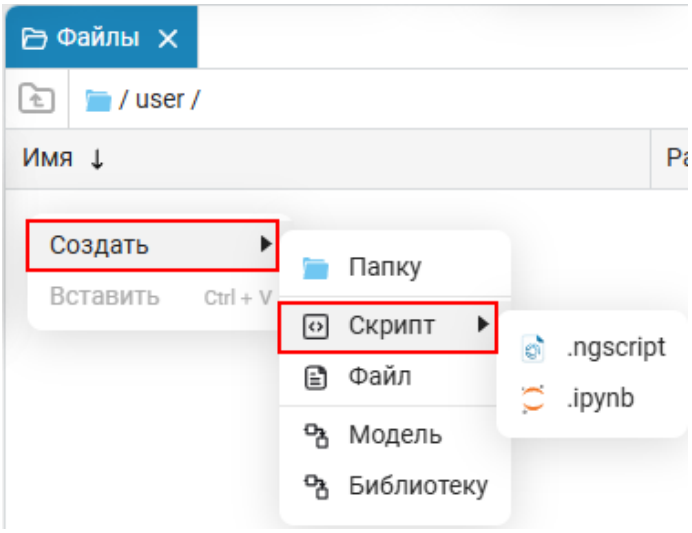
Также можно кликнуть на **Файловый браузер** и последовательно выбрать команды **Создать → Скрипт** (рис. 3.1, б).

По умолчанию скрипты **Engee** имеют формат ngscript, однако можно работать и с форматом jl и ipynb:

- jl — формат скриптов на языке Julia. Редактор скриптов поддерживает рефакторинг и запуск скриптов в этом формате;
- ipynb — универсальный формат написания скриптов на языке Python.



а)



б)

Рисунок 3.1 – Запуск редактора скрипта

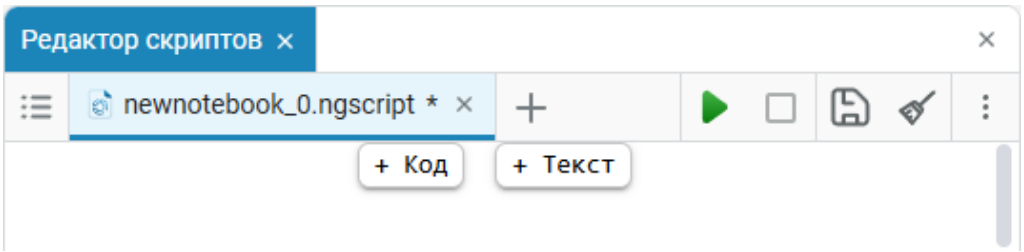
Ячейки скрипта. Интерактивный скрипт состоит из ячеек — отдельных блоков, которые можно запускать независимо (рис. 3.2, а).

Ячейки бывают двух типов: кодовые и текстовые.

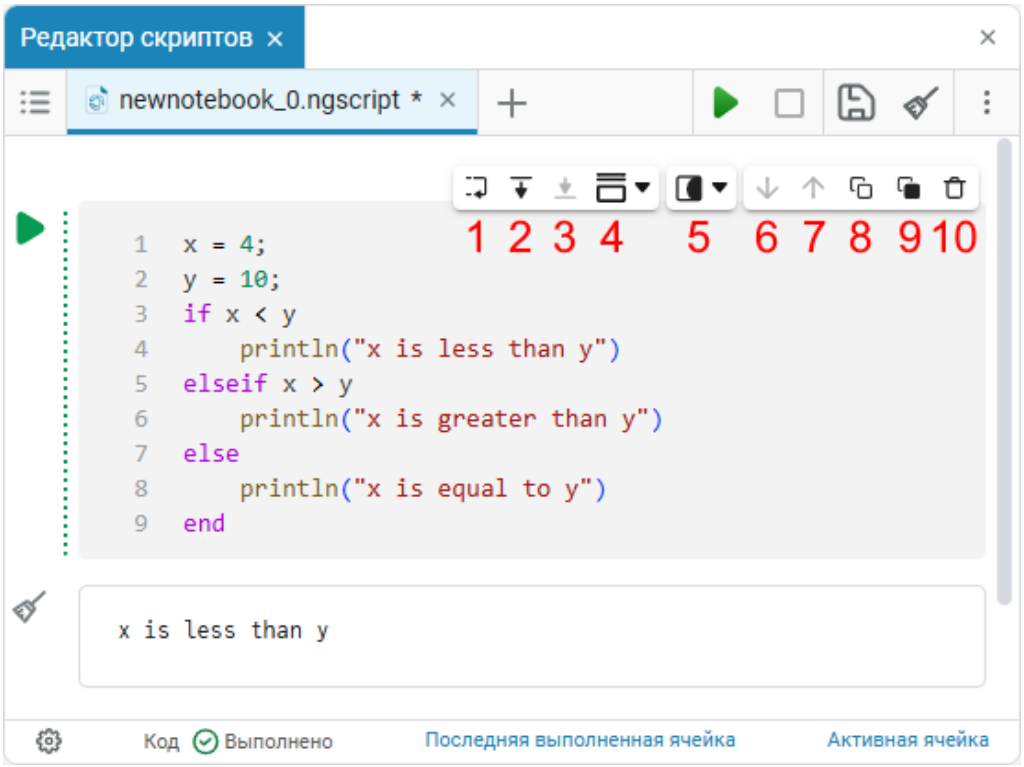
Кодовые ячейки — предназначены для работы с кодом. Такие ячейки можно запускать по одной, выполнять только выделенный фрагмент или выполнять весь код целиком (рис. 3.2, б).

Текстовые ячейки — для пояснений, формул и визуального оформления. Поддерживаются Markdown, LaTeX и HTML (рис. 3.2, в).

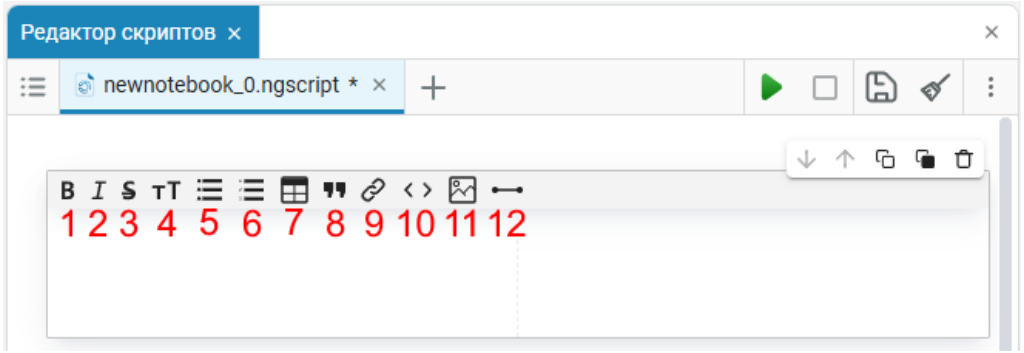
Новые секции можно добавлять в любом месте рабочей области редактора скриптов (например, в начале, конце или между двумя другими секциями), кликнув по всплывающим кнопкам **+Код** или **+Текст** (рис. 3.2, а).



а)



б)



в)

Рисунок 3.2 – Секции редактора скриптов:
а – общий вид; б – кодовая секция; в – текстовая секция

Интерфейс кодовой секции содержит иконки, отвечающие за следующие команды (см. рис. 3.2,б):

1. **Выполнить и перейти** — выполняет код в данной секции.
2. **Выполнить до конца** — выполняет все секции с кодом ниже выбранной секции.
3. **Выполнить до ячейки** — выполняет все секции с кодом выше выбранной секции.
4. **Поменять положение результата работы кода** — выбирает позицию для вывода. Имеет два варианта:

4.1. **Вывод результата справа** — выводит результат справа для конкретной секции.

4.2. **Спрятать вывод результата** — прячет выбор результата для конкретной секции.

5. **Маска** — добавляет маску для выбранной кодовой ячейки (подробнее см. Маски кодовых ячеек [2]).
6. **Переместить ниже** — перемещает секцию на одну вниз.
7. **Переместить выше** — перемещает секцию на одну вверх.
8. **Копировать** — копирует секцию.
9. **Вставить** — вставляет секцию.
10. **Удалить** — удаляет секцию.

В кодовой ячейке также можно:

- выполнить выделенный участок кода (кликнув правой кнопкой мыши по выделенной строке или — горячая клавиша Shift+F7 [2]);
- выполнить текущую строку с курсором (Ctrl+Shift+F5 на Windows/Linux или Shift+⌘+F5 на macOS).

Интерфейс текстовой секции включает команды (см. рис. 3.2, в):

1. **Полужирный** — выделяет текст полужирным шрифтом.
2. **Курсив** — делает выделенный текст курсивом.
3. **Зачеркнутый** — зачеркивает выделенный текст.
4. **Показать/скрыть заголовок** — добавляет заголовок с помощью знака решетки # или удаляет его.
5. **Добавить маркированный список** — добавляет два пункта маркированного списка.
6. **Добавить нумерованный список** — добавляет два пункта нумерованного списка.
7. **Вставить таблицу** — добавляет таблицу. Таблица может быть отредактирована в поле ввода секции.
8. **Вставить цитату** — вставляет шаблон для цитаты.
9. **Вставить ссылку** — вставляет шаблон для ссылки.
10. **Вставить секцию с кодом** — вставляет шаблон для кода. Данный код не может быть выполнен, поскольку не задан в секции с кодом и служит только для визуального восприятия.

11. **Вставить картинку** — открывает файловую систему вашего компьютера для выбора изображений.

12. **Вставить горизонтальную линию** — вставляет горизонтальную разделительную линию.

Аналогично секциям кода в текстовых ячейках использует следующие команды: перемещение, копирование, вставка и удаление.

Скрытие ячеек.

В редакторе скриптов Engee можно скрыть:

- текстовые ячейки с заголовками более низкого уровня;
- кодовые ячейки (имеют наименьший уровень заголовка);
- текстовые ячейки без заголовков (также наименьший уровень);

Для скрытия ячеек сначала необходимо оформить текстовую ячейку. Текстовая ячейка должна быть отмечена как параграф в формате Markdown [2] и иметь соответствующий уровень и расположение:

- ячейка должна содержать заголовок от 1 до 6 уровня (#, ## и т.д.);
- под заголовком верхнего уровня должна находиться ячейка с заголовком более низкого уровня (рис. 3.3).

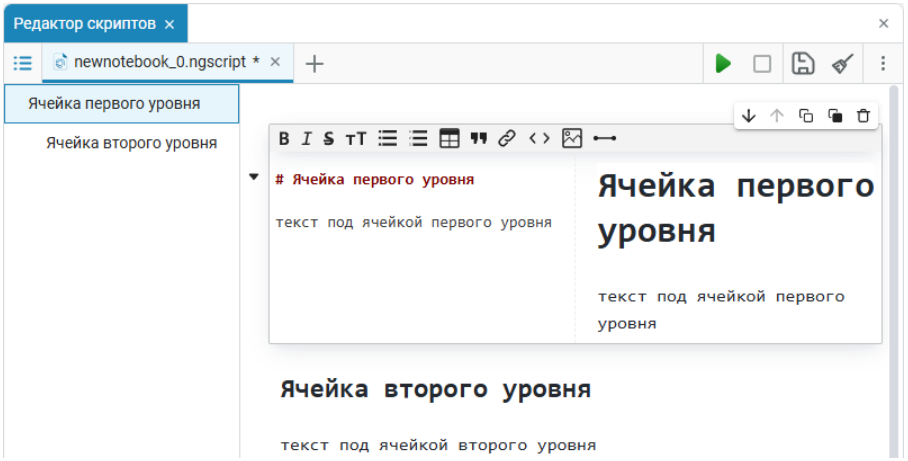


Рисунок 3.3 – Оформление уровней текстовых ячеек в формате Markdown

Если у текстовой ячейки с заголовком первого уровня под ней находится, например, 10 кодовых ячеек — это образует одну секцию, которая будет скрыта. Когда появляется 11-я ячейка — новый параграф с заголовком первого уровня или такого же уровня, что и первая текстовая ячейка, то начинается новая секция.

Заголовки текстовых ячеек с первого по шестой уровень образуют **Содержание** (рис. 3.4).

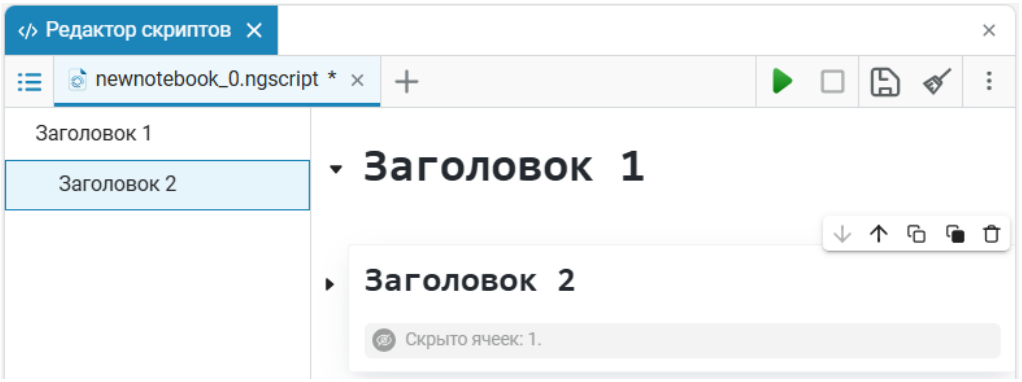
Нажав три точки меню в разделе содержания, можно:

- Добавить вложенный параграф + — добавляет текстовую ячейку с заголовком на уровень выше. Например, если вложенный параграф добавляется к текстовой ячейке первого уровня (#), то он станет

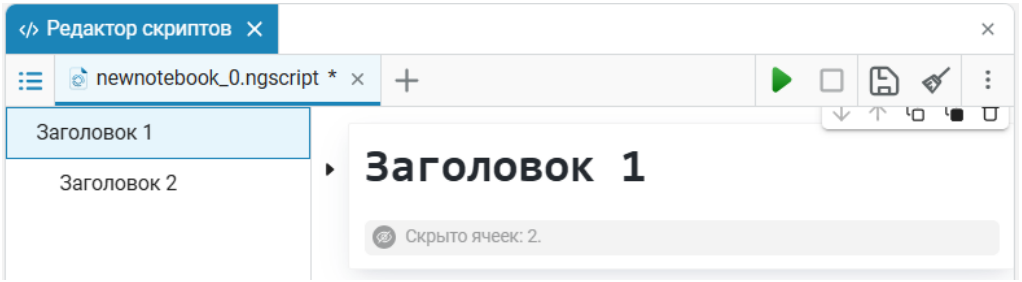
[2] Markdown — облегчённый язык разметки, инструмент преобразования текста в HTML.

ячейкой второго уровня (##). При достижении максимальной вложенности (последняя текстовая ячейка имеет заголовок шестого уровня) — кнопка недоступна.

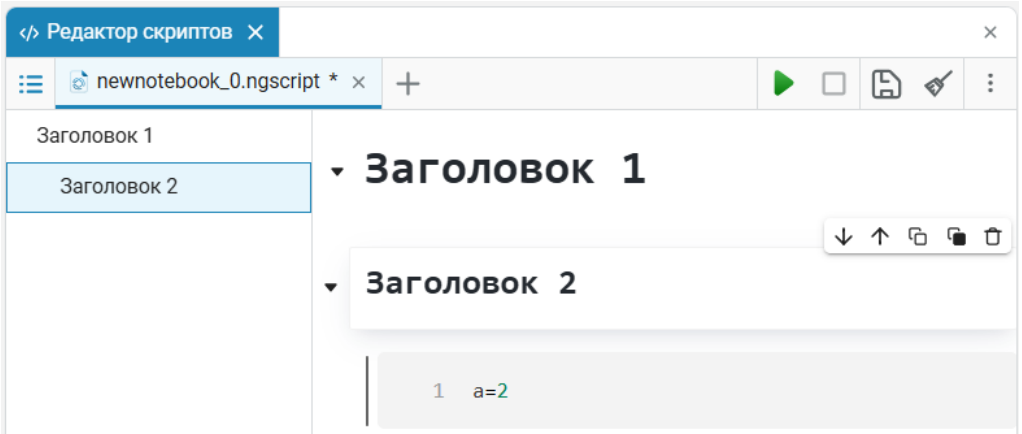
- Запустить параграф — запускает исполнение всех ячеек с кодом выбранного параграфа. Если параграф не содержит кодовых ячеек — кнопка недоступна.
- Удалить параграф — удаляет выбранный параграф. Заголовки и ячейки с кодом, вложенные в секцию параграфа, удалены не будут.



а)



б)



в)

Рисунок 3.4 – Скрытые (а, б) и открытые (в) ячейки

Запуск и остановка скриптов. Для запуска или остановки скрипта необходимо нажать кнопку **Пуск** или **Стоп**. Кнопки находятся на панели запуска скрипта (рис. 3.5).

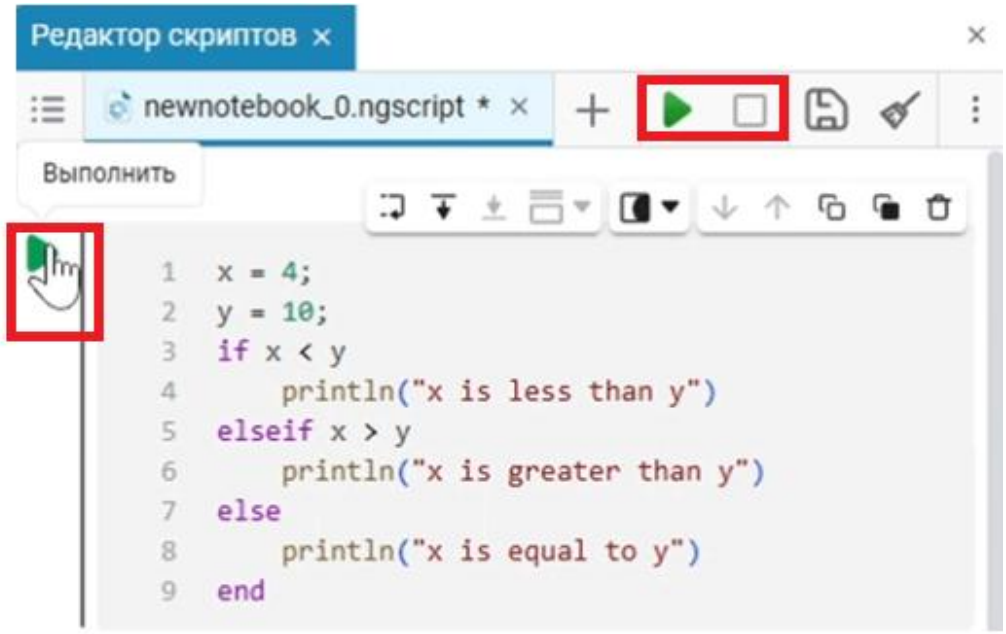


Рисунок 3.5 – Запуск всего скрипта (вверху) или секции (слева)

Отдельную секцию можно запустить или остановить с помощью аналогичных кнопок, расположенных с левой стороны секции кодовой ячейки (рис. 3.5, 3.6).

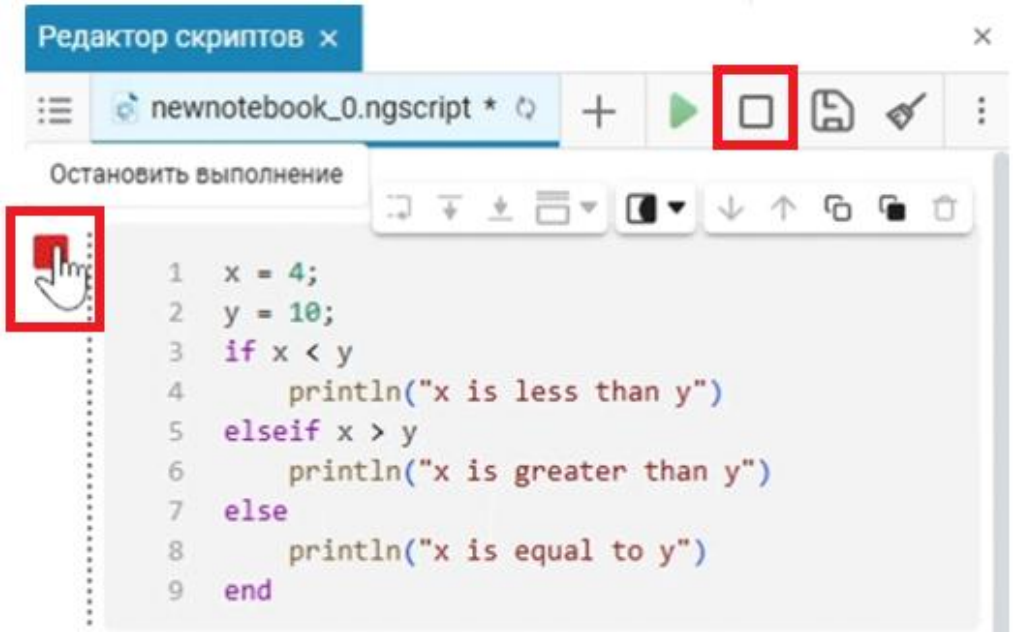


Рисунок 3.6 – Остановка скрипта (вверху) или секции (слева)

Очистить результаты выполнения скрипта, которые появляются непосредственно под ним, можно с помощью специальной кнопки, как показано на рис. 3.7.

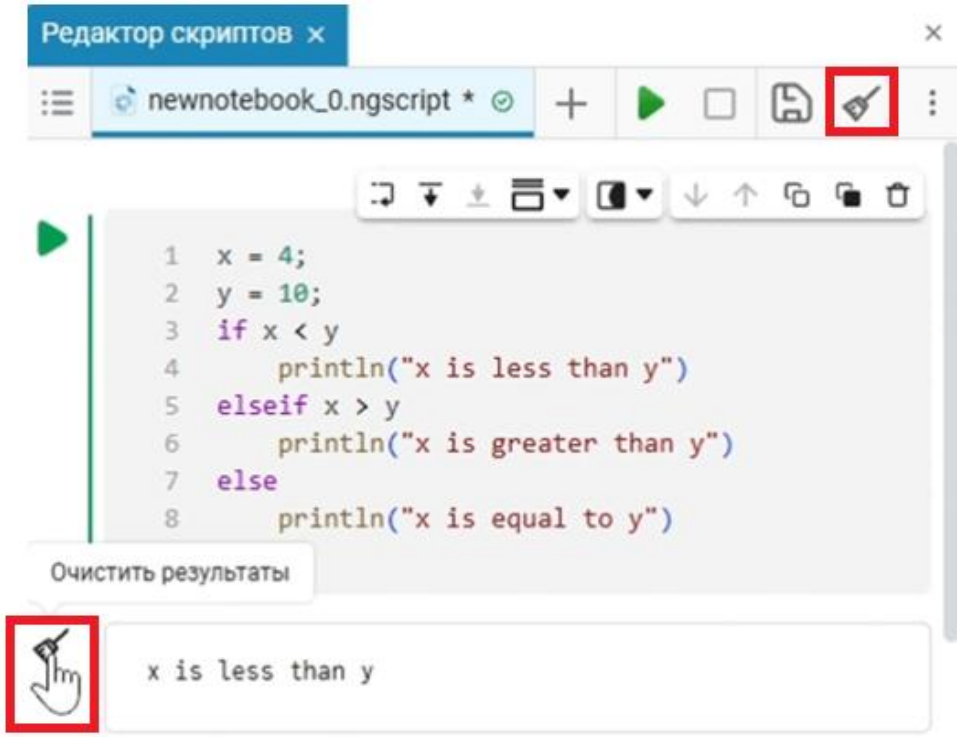


Рисунок 3.7 – Очистка результатов скрипта (вверху) или секции (слева)

В случае если вывод исполнения кода занимает слишком много места, его можно свернуть в более компактную форму. Для этого служит кнопка **Ограничить вывод** . Чтобы вернуть вывод в прежнее состояние используется кнопка **Раскрыть вывод** .

Редактор является многооконным. Окно каждой программы оформляется как вкладка, окно нового скрипта можно создать, нажав на значок + (см рис. 3.2 – 3.7).

Если планируется открытие скриптов, написанных в Engee, в сторонних системах (например, Jupyter), то рекомендуется изменить расширение сохраненного скрипта при помощи файлового браузера Engee (рис. 3.8).

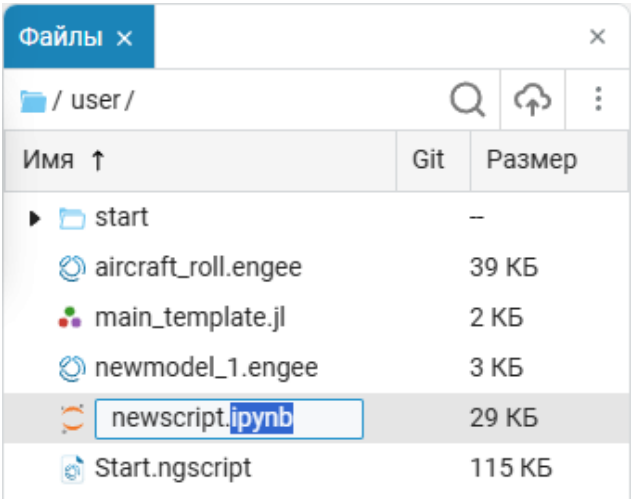


Рисунок 3.8 – Изменение расширения файла скрипта

Редактор выполняет синтаксический контроль программного кода по мере ввода текста. При этом используются следующие цветовые выделения:

- **операторы и переменные –** черный цвет;
- **синтаксические ошибки –** красный, черный и серый цвет;
- **символьные переменные (в апострофах) –** красно-коричневый цвет;
- **функции –** серо-зеленый цвет;
- **константы, комментарии после знака # –** зеленый цвет;
- **скобки –** синий цвет;
- **ключевые слова языка программирования –** сиреневый цвет.

Благодаря цветовым выделениям вероятность синтаксических ошибок снижается.

Семантические ошибки, связанные с неверным применением операторов или функций (например, применение оператора «-» вместо «+» или функции cos(x) вместо sin(x) и т. д.), не способна обнаружить ни одна система программирования.

Файл скрипта по умолчанию имеет название newnotebook_0. Чтобы дать ему другое имя, необходимо, нажав правой кнопкой мыши на имени файла, выполнить команду «Сохранить как» и в диалоговом окне указать папку и имя этого файла.

Основные особенности записи кода:

- обычно каждый оператор записывается в отдельной строке текста программы. Признаком конца оператора является символ (он не появляется в окне) возврата каретки и перехода на следующую строку, который вводится в программу при нажатии клавиши [Enter];
- можно размещать несколько операторов в одной строке. Тогда предыдущий оператор должен заканчиваться символом точка с запятой «**;**»;
- длинный оператор можно записывать в несколько строк. При этом предыдущая строка оператора должна заканчиваться тремя точками "…";
- если оператор не заканчивается символом «;», результат его действия при выполнении программы будет выведен;
- строка программы, начинающаяся с символа «#» не выполняется, эта строка воспринимается системой как комментарий;
- переменные не описываются и не объявляются;
- использование матрицы или переменной, в которой предварительно не введены или не вычислены значения её элементов, приведет к сообщению «EngeeNothing()».

Комментарий – один из базовых элементов языка программирования – не является лексемой. Внутри комментария можно использовать любые допустимые на данном компьютере символы, а не только символы из алфавита языка программирования, поскольку компилятор комментарии игнорирует.

Они вводятся с помощью символа «#», например:

```
# функция вычисления……
```

Комментарии могут ограничиваться парами символов #= и =#, в этом случае можно организовать многострочный комментарий.

3.2.2. Функции

Функция – это отдельная программа, которая обычно служит для описания каких-либо стандартных действий, требующих повторения. Многократно может вызываться из основной программы. В языке Julia функция представляет собой объект, который сопоставляет кортеж значений аргументов с возвращаемым значением. Базовый синтаксис определения функций в Julia следующий:

```
function f(x,y)
    x + y
end
```

После ключевого слова «function» через пробел указывается имя функции, а в скобках – ее аргументы, в примере имя функции f, а ее аргументы x и y. Далее в теле функции записывается выражение, в примере это x + y, вся конструкция заканчивается ключевым словом «end». Выражений может быть несколько. Функция возвращает значение последнего вычисленного выражения.

В Julia имеется более сжатый синтаксис для определения функции. Традиционный синтаксис объявления функций, показанный выше, эквивалентен следующей компактной форме:

```
f(x,y) = x + y
```

В этой форме присваивания тело функции должно быть одним выражением, но может быть и составным выражением (см. п. 3.2.3).

Вызывается функция с помощью употребления ее имени вместе со списком значений параметров, взятым в круглые скобки (рис. 3.9).

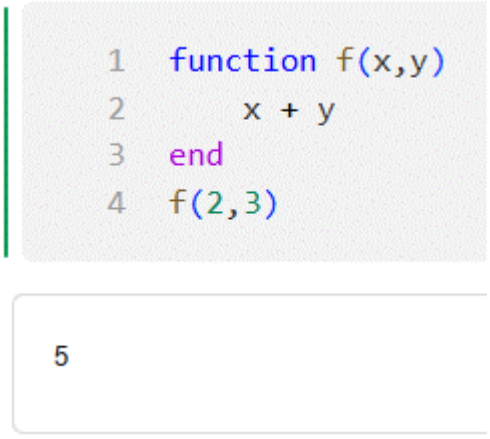


Рисунок 3.9 – Результат работы функции

3.2.3. Линейные алгоритмы

Линейный алгоритм – набор команд, выполняемых последовательно во времени друг за другом.

При составлении линейных программ с помощью интерактивных скриптов одним из основных операторов является оператор присваивания.

Общий вид оператора присваивания:

Имя_переменной = Выражение

где **Имя_переменной** – имя простой переменной, структурированной переменной (вектора, матрицы), имя функции; **Выражение –** арифметическое, логическое или строковое выражение.

Тип переменной определяется системой автоматически по типу выражения, поэтому нет необходимости при программировании следить за соответствием типов данных в операторе присваивания. Если выражение содержит и арифметические и строковые элементы, то переменная будет численной, т.е. предпочтение отдается числовому типу данных.

Например,

```
A = cos(pi/3)+a-b^2*c^2+7.87  # алгебраическое выражение
n = "ответ"; a = 'x'           # символьные выражения
R = (x>5)&(x<=10)              # логическое выражение
```

3.2.4. Составные выражения

В случае если необходимо использовать выражение, в котором несколько подвыражений вычисляются по порядку и которое возвращает результат последнего подвыражения, в Julia можно применить две конструкции: блоки begin-end и цепочки через знак «;». Значением обеих конструкций составных выражений является результат последнего подвыражения (рис. 3.10).

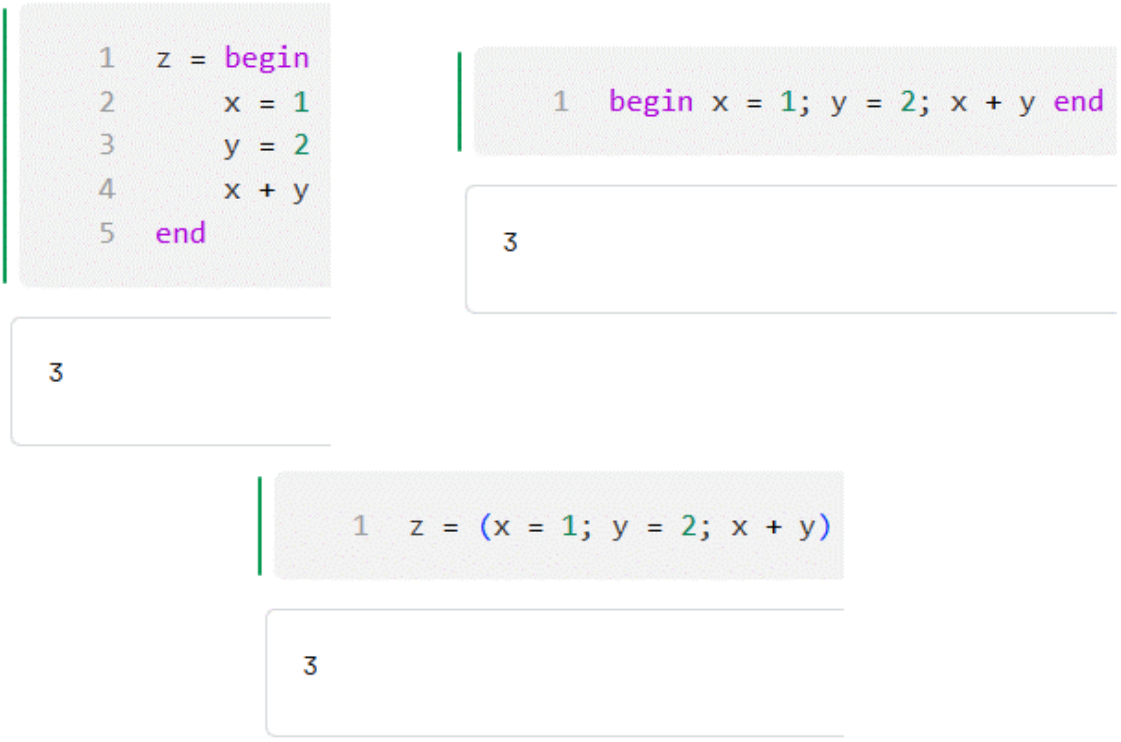


Рисунок 3.10 – Составные выражения

3.2.5. Организация диалога с пользователем

При программировании часто возникает необходимость вводить исходные данные не с помощью оператора присваивания, а с помощью операторов ввода в диалоговом режиме. Преимущества такого ввода очевидны: при вводе новых исходных данных нет необходимости вносить изменения в программу.

Функции ввода.

Ввод данных с клавиатуры можно осуществить с помощью функции `readline()`.

Если нужно ввести переменную `x` строкового типа, то используется следующая конструкция:

```
x = readline()
```

Функция `readline` предназначена для ввода данных только строкового типа.

Если нужно ввести переменную простого числового типа (целого, вещественного или логического), то дополнительно используется функция `parse`:

```
x = parse(Type, readline())
```

Ее первый аргумент `Type` задает тип переменной (Int, Float64, Float32, Bool и т.д.), в который нужно преобразовать строковую переменную, заданную вторым аргументом `readline()`.

Рисунок 3.11 – Примеры работы функций ввода

При запуске на выполнение программы, содержащей функции ввода следует учитывать, что пока пользователь не ввел данные в ответ на запрос

своей программы, функция ввода продолжает свою работу. Примеры работы функций ввода приведены на рис. 3.11.

Функции вывода.

Если необходимо вывести данные на экран в произвольном месте кодовой ячейки, то это можно сделать с помощью следующих функций:

- `print` – вывод на экран значений переменных, констант, математических или логических выражений. Если нужно вывести несколько значений `(x, y, z)`, то они перечисляются через запятую: `print(x, y, z);`
- `println` – то же, что и `print`, но с переходом на новую строку, т.е. следующий вывод будет производиться в новой строке;
- `display` – то же, что и `print`, но у этой функции может быть только один аргумент (константа, переменная или выражение).
- `show` – то же, что и `display`.

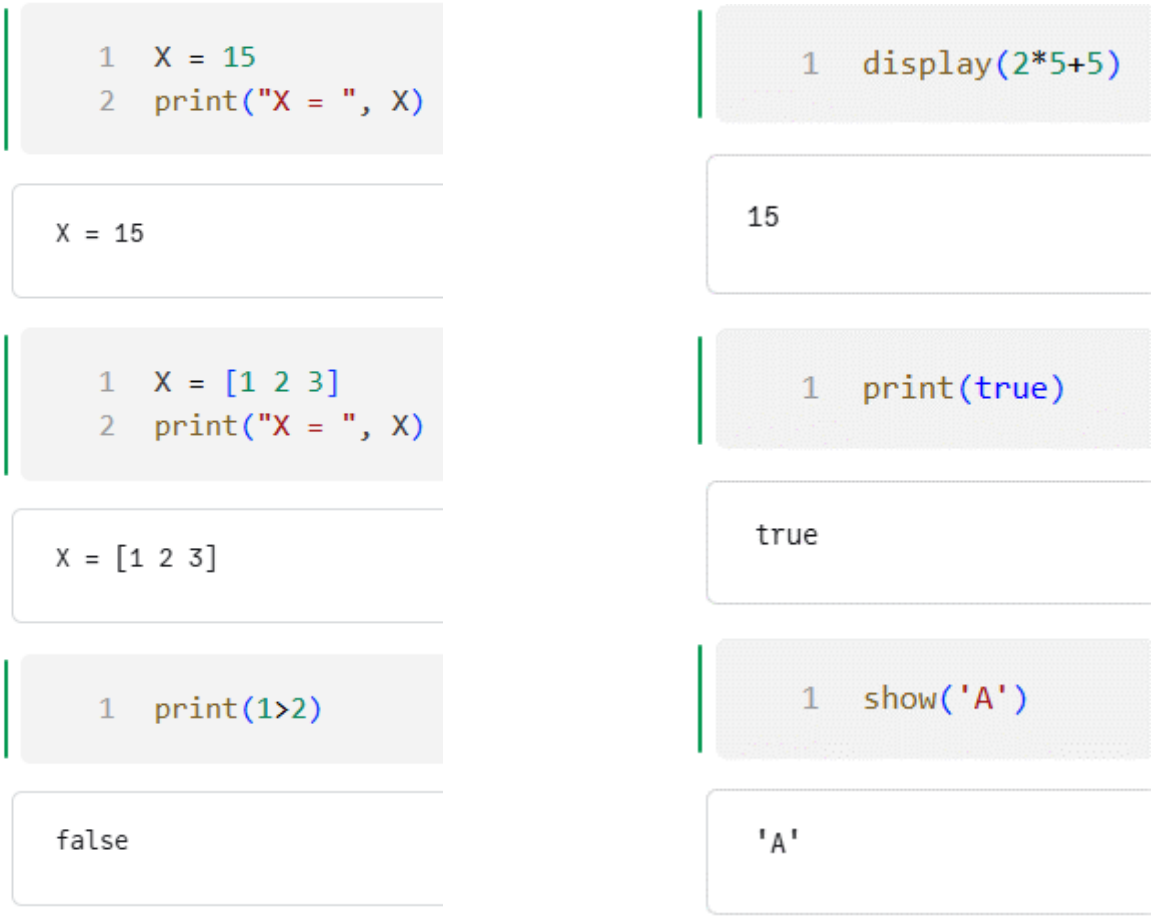


Рисунок 3.12 – Примеры работы функций вывода

3.2.6. Разветвляющиеся алгоритмы

Разветвляющийся алгоритм – алгоритм, содержащий хотя бы одно условие, в результате проверки которого обеспечивается переход на один из двух возможных шагов.

С помощью разветвляющих алгоритмов можно реализовывать логику выполнения операций и создавать повторяющиеся (итерационные,

рекуррентные) вычисления. В таких алгоритмах используют логические операции и операции отношения.

3.2.7. Отношения и логические операции

Операции отношения сравнивают два числа, выдавая ответ истинна или ложь (true/false). Логические операции рассматривают true/false утверждения и выполняют действия, которые соответствуют true- или false-операторам. Отношения и логические операции используются в математических выражениях и в комбинациях с другими командами.

Стандартно имеется шесть операций отношения, которые приведены в табл. 3.1.

Таблица 3.1 – Операции отношения

Операторы отношения	**Название**
==	Равно =
!=	Не равно ≠
<	Меньше <
<=	Меньше или равно ≤
>	Больше >
>=	Больше или равно ≥

Основные логические операции приведены в табл. 3.2.

Таблица 3.2 – Логические операции

Оператор	**Выражение**	**Название**	**Описание**
!	!x	НЕ	Возвращает true, если операнд x имеет значение false и false, если операнд x есть true
&	x & y	И (побитовая операция)	Если оба операнда (x и y) равны true, то и результат true. В противном случае false
&&	x && y	И (вычисляемое по сокращенной схеме)	Будет иметь значение false, если x равно false, независимо от значения y. Подвыражение y вычисляется только в том случае, если x равно true.
|	x | y	ИЛИ (побитовая операция)	Если один или второй операнд true, то и результат true. В противном случае false.
||	x || y	ИЛИ (вычисляемое по сокращенной схеме)	Будет иметь значение true, если x равно true, независимо от значения y. Подвыражение y вычисляется только в том случае, если x равно false

Приоритет логических операций следующий:

&,| – самый высокий приоритет;

&& – средний приоритет;

|| – самый низкий приоритет.

3.2.8. Операторы управления вычислительным процессом

Условный оператор `if`.

I. В самом простом случае (рис. 3.13) синтаксис оператора `if` имеет вид:

```
if <выражение>
<операторы>
end
```

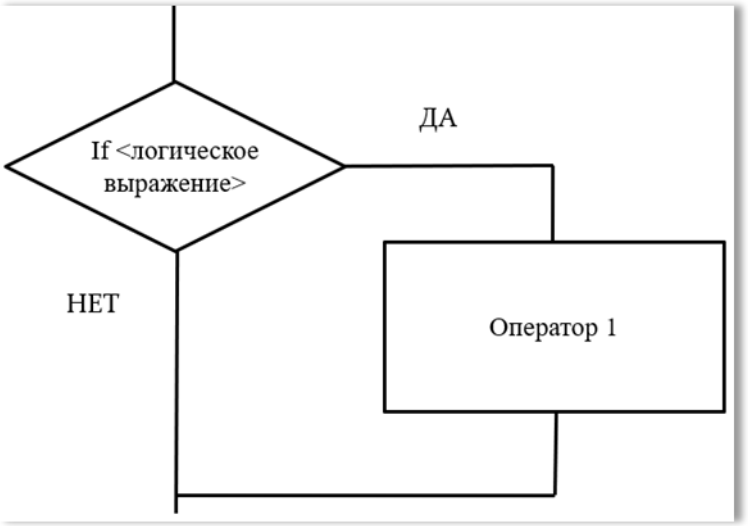


Рисунок 3.13 – Блок-схема условного оператора if

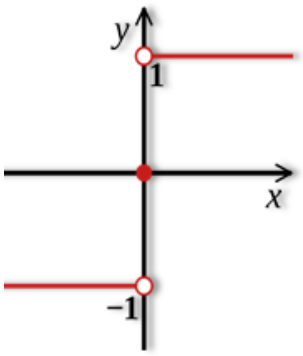
Если значение параметра "выражение" соответствует значению true, то выполняется оператор, иначе он пропускается программой. Следует отметить, что "выражение" является условным выражением, в котором выполняется проверка некоторого условия при помощи операторов отношения, приведённых в табл. 3.2.

Приведём пример реализации функции **sgn(x)** (сигнум, знак)[3], которая возвращает +1, если число больше нуля, -1 – если число меньше нуля и 0, если число равно нулю. Зададим значение переменной x равным 5. Очевидно, функция должна возвратить 1 (рис. 3.14).

В приведённом примере видно, что все эти три условия являются взаимоисключающими, т.е. при срабатывании одного из них нет необходимости проверять другие. Реализация именно такой логики позволит увеличить скорость выполнения программы.

---

[3] Кусочно-постоянная функция действительного аргумента: $sng(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$




```
x=5;
 if x>0
    print(1);
end
if x<0
    print(-1)
end
if x==0
    print(0);
end
```

```
1
```

```
function Sng(x)
x=5;
 if x>0
    print(1);
end
if x<0
    print(-1)
end
if x==0
    print(0);
end
end
Sng(5) #вызов функции
```

```
1
```

а) б) в)

Рисунок 3.14 – График (а) и программа реализации функции sgn(x) (б,в)

Эта же простая программа может быть записана в виде функции (рис. 3.14, в). Чтобы вызвать функцию достаточно в командном окне или в скрипте указать имя функции и параметр в скобках, например, Sgn(5). В этом случае система выдаст ответ, также равный 1 (рис. 3.14, в, рис. 3.15).

```
engee> Sng(5)
1
engee>
```

Рисунок 3.15 – Вызов функции в командном окне

Если функция вызывается из скрипта, то для исполнения программы необходимо нажать на зеленую кнопку слева.

В Julia также имеется встроенная функция **sign(z),** которая определяет знак числа **z.**

II. Второй вариант оператора `if` использует зарезервированное слово `else` (рис. 3.16)

```
if <выражение>
<операторы1>    # выполняются, если условие истинно
else
<операторы2>    # выполняются, если условие ложно
end
```

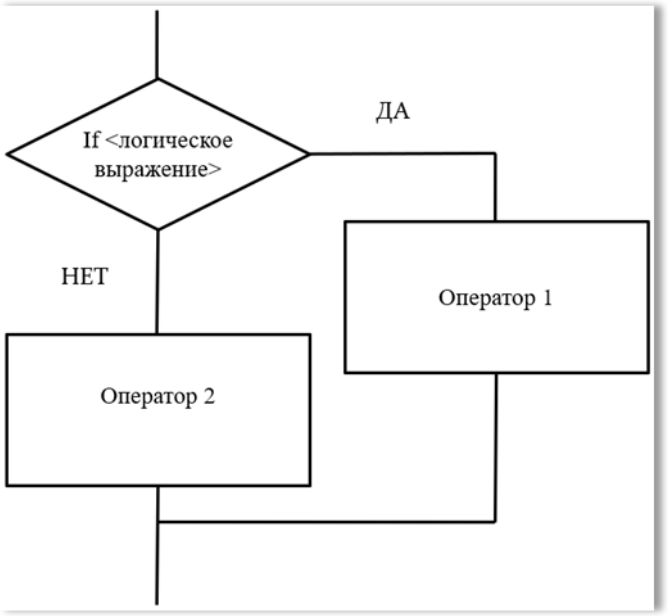


Рисунок 3.16 – Блок-схема оператора if else

Тогда приведённый выше пример можно записать следующим образом:

```
x = 5;
if x > 0
     print(1);
else
     if x < 0
          print (-1);
     else
          print (0);
     end
end
```

```
x = 5;
if x > 0
    print(1);
else
if x < 0
    print(1);
else
    print(1);
end
end
```

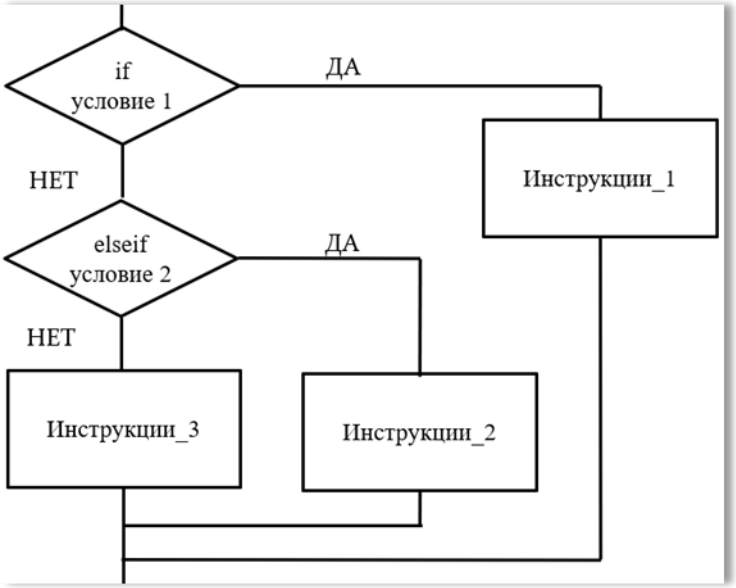
```
1
```

Рисунок 3.17 – Программа реализации функции sign оператором if-else

Следует обратить внимание на отступы при написании кода: операторы не записывают один под другим. Это традиционно делается для лучшей читаемости программы, т.е. при такой записи видно, где начинается и заканчивается определенная конструкция команд. Хотя строгой необходимости в этом нет. В нашем примере имеется встроенная функция if-else в другую такую же функцию.

III. Еще одна конструкция оператора `if` использует конструкцию `elseif` (рис. 3.18):

```
if <выражение1>
<операторы1>      # выполняются, если истинно выражение1
elseif <выражение2>
<операторы2>  # выполняются, если истинно выражение2
...
elseif <выражениеN>
<операторыN>  # выполняются, если истинно выражениеN
end
```



а)

```
x = 5;
if x > 0
    print(1);  #выполняется, если x > 0
elseif x < 0
    print(-1); #выполняется, если x < 0
else
    print(0);  #выполняется, если x = 0
end
```

```
1
```

б)

Рисунок 3.18 – Блок-схема оператора if-elseif (а)
и программа реализации функции sgn оператором if-elseif (б)

Оператор цикла `while`.

С помощью операторов цикла, например, выполняется программирование рекуррентных алгоритмов, подсчёта суммы ряда, перебора элементов массива и многое другое.

В самом простом случае цикл в программе организуется с помощью оператора `while`, который имеет следующий синтаксис:

```
while <условие>
      <операторы>
end
```

Здесь <условие> означает условное выражение подобное тому, которое применяется в операторе `if`, и цикл `while` работает до тех пор, пока это условие **true**. Если условие будет **false** до начала выполнения цикла, то операторы, входящие в цикл, не будут выполнены ни разу.

Например, запишем программу для подсчёта суммы ряда $S = \sum_{i=1}^{6} i$ (это так называемое треугольное число – сумма первых n (в данном случае n=6) натуральных чисел, равная количеству точек, из которых можно построить правильный треугольник):

```
S = 0;          #начальное значение суммы
i=1;            #счётчик суммы
while i <= 6    #цикл (работает пока i <= 6)
    S=S+i;      #подсчитывается сумма
    i=i+1;      #увеличивается счётчик на 1
end             #конец цикла
print(S);       #вывод суммы на экран
```

```
21
```

Рисунок 3.19 – Программа подсчета треугольного числа (сумма натуральных чисел)

Теперь пусть значение суммы ограничено, например, числом 10. Тогда в операторе цикла получается два условия: либо счетчик по i доходит до 6, либо значение суммы S не превысит 10. Данную логику можно реализовать с помощью составного условного выражения (оператор &&) в операторе цикла `while`:

```
S = 0;                      #начальное значение суммы
i=1;                        #счётчик суммы
while i <= 6 && S < 10      #цикл (работает пока i <= 6 и S < 10)
    S=S+i;                  #подсчитывается сумма
    i=i+1;                  #увеличивается счётчик на 1
end                         #конец цикла
print(S);                   #вывод суммы на экран
```

```
10
```

Рисунок 3.20 – Программа подсчета суммы натуральных чисел с ограничением значения суммы

Оператор цикла `for`.

Синтаксис оператора **for** имеет следующий вид:

```
for <счётчик> = <нач. значение>:<шаг>:<кон. значение>
      <операторы цикла>
end
```

Рассмотрим работу данного цикла на примере реализации алгоритма поиска максимального значения элемента в векторе (рис. 3.21).

```
A = [5 3 7 4 2 9 5 4 1 2];
m = A[1];      #текущее максимвльное значение (равно значению 1-го первого элемента)
for i=2:length(A) #цикл от 2 до конца вектора с шагом 1 (по умолчанию)
    if m < A[i]   #если A(i) < m,
        m = A[i]; #то m = A(i)
    end
end               #конец цикла for
print(m)
```

```
9
```

Рисунок 3.21 – Программа поиска максимального элемента вектора

Если величина шага не указывается явно, то он берётся по умолчанию равным 1. Для реализации цикла со счётчиком от большего значения к меньшему, нужно явно указывать шаг, например, -1. Если этого не сделать, то цикл сразу завершит свою работу.

В общем случае в цикле `for` возможна итерация по любому контейнеру. В таких случаях в качестве полностью равносильной альтернативы символу «=» обычно применяется ключевое слово `in` или ∈, что делает код более понятным.

```
for i in [1,4,0]
    println(i)
end
```

```
1
4
0
```

```
for s ∈ ["один","четыре","ноль"]
    println(s)
end
```

```
один
четыре
ноль
```

Рисунок 3.22 – Примеры оформления итерации в цикле for

Приведём еще пример. На рис. 3.23 приведена программа, которая преобразует случайную матрицу в диагональные матрицы, главные диагонали которых составлены соответственно из главной и побочной диагонали исходной матрицы. Результат работы программы приведен на рис. 3.24.

```
# Создание случайной матрицы 6x6, содержащей числа от 1 до 9
A = rand(1:9, 6, 6)
#= Задание нулевых матриц с целым типом данных (Int),
в которые будут записаны диагонали =#
DIAG1 = zeros(Int, 6, 6)
DIAG2 = zeros(Int, 6, 6)
# Создание диагональных матриц
for i in 1:6
    DIAG1[i, i] = A[i, i]     #с главной диагональю из A
    DIAG2[i, 7-i] = A[i, 7-i] #с побочной диагональю из A
end
# Вывод результатов
println("Исходная матрица:")
display(A)
println("Диагональная матрица (главная диагональ):")
display(DIAG1)
println("Диагональная матрица (побочная диагональ):")
display(DIAG2)
```

Рисунок 3.23 – Программа построения диагональных матриц

```
Исходная матрица:

6×6 Matrix{Int64}:
 8  5  9  7  7  8
 4  5  9  1  1  4
 1  9  9  2  6  7
 9  4  6  7  2  2
 7  3  4  9  1  3
 1  1  4  3  4  6

Диагональная матрица (главная диагональ):

6×6 Matrix{Int64}:
 8  0  0  0  0  0
 0  5  0  0  0  0
 0  0  9  0  0  0
 0  0  0  7  0  0
 0  0  0  0  1  0
 0  0  0  0  0  6

Диагональная матрица (побочная диагональ):

6×6 Matrix{Int64}:
 0  0  0  0  0  8
 0  0  0  0  1  0
 0  0  0  2  0  0
 0  0  6  0  0  0
 0  3  0  0  0  0
 1  0  0  0  0  0
```

Рисунок 3.24 – Результат работы программы построения диагональных матриц

3.3. Порядок выполнения работы

3.3.1. Изучить теоретические сведения. Выполнить тренировочные задания, повторив операции, представленные на рис. 9-24.

3.3.2. Выполнить задание согласно выданному преподавателем варианту. Задание оформить в виде интерактивного скрипта, файл скрипта сохранить в системе Engee и/или на внешний накопитель.

Варианты к заданию 3.2 указаны в таблице 3.3.

Таблица 3.3 – Варианты к заданию 3.3.2

№ варианта	Задание
1.	Написать функцию, которая определяет знак многочлена ax^3-bx^2-cx+d в точке, значение которой задается с клавиатуры. Параметры многочлена передаются в функцию в качестве входных параметров.
2.	Написать программу, которая находит количество элементов последовательности целых чисел, введенной пользователем, кратных числу 2 и кратных числу 3.
3.	Написать функцию, которая вычисляет сумму ряда $\sum_{i=0}^{n}\frac{-1^{\,i}\;i+1}{i!}$ для заданного n.
4.	Написать скрипт, который находит количество четных элементов последовательности, введенной пользователем.

№ варианта	Задание
5.	Создать функцию, которая определяет, принадлежит ли число, заданное с клавиатуры, массиву чисел, который передается в функцию как параметр.
6.	Написать функцию, которая многократно вычисляет длину окружности по заданному пользователем радиусу. Остановка программы должна происходить при введении отрицательного значения радиуса.
7.	Создать функцию, которая определяет, содержит ли строка, введенная с клавиатуры, пробелы. Вывести результат в виде строки.
8.	Написать программу, которая находит минимальный элемент последовательности целых чисел, введенной пользователем.
9.	Написать функцию, которая определяет знак дискриминанта квадратного уравнения.
10.	Написать функцию, которая вычисляет сумму ряда $\sum_{i=0}^{n} \frac{-1^{i+1} i^2}{i+1}$ для заданного n.
11.	Написать программу, которая находит номер максимального элемента последовательности целых чисел, введенной пользователем.
12.	Написать функцию, которая вычисляет количество элементов последовательности целых чисел, введенной пользователем, кратных введенному числу k.
13.	Написать программу, которая вычисляет сумму элементов последовательности целых чисел, введенной пользователем, с четными номерами.
14.	Написать функцию, определяющую корни квадратного уравнения.
15.	Написать функцию, которая вычисляет сумму элементов главной диагонали введенной пользователем квадратной матрицы.
16.	Написать функцию, которая вычисляет площадь равностороннего треугольника с основанием a.
17.	Написать программу, которая заменяет первый столбец квадратной матрицы, введенной пользователем, на столбец, в котором имеется максимальный элемент матрицы.
18.	Написать функцию, которая вычисляет произведение всех положительных элементов последовательности целых чисел, введенной пользователем.
19.	Написать программу, вычисляющую значения $y(x) = x^2$ для всех x на отрезке от 1 до 10 с шагом равным 3.
20.	Написать функцию, которая составляет векторы из n нечетных чисел натурального ряда.
21.	Написать функцию, которая вычисляет факториал числа n.
22.	Написать программу, находящую среднее арифметическое элементов заданного вектора и заменить первый элемент этим значением.
23.	Написать функцию, которая заменяет максимальный элемент заданного вектора средним значением всех его элементов.

№ варианта	Задание
24.	Написать программу, заменяющую элемент заданной матрицы с индексами (1,1) произведением всех элементов матрицы.
25.	Написать функцию, которая складывает все элементы заданной матрицы, кроме элементов главной диагонали.
26.	Написать программу, которая находит строку квадратной матрицы, введенной пользователем, с наименьшей суммой элементов, и заменяет все элементы этой строки нулями.
27.	Написать функцию, которая вычисляет объем правильной треугольной пирамиды (тетраэдра) с длиной ребра a.
28.	Написать программу, которая находит в заданном векторе первый отрицательный элемент и заменяет его значением модуля этого элемента.
29.	Написать функцию, которая создает квадратную матрицу заданного размера n x n, где все элементы главной диагонали равны 2, а все остальные элементы равны 0.
30.	Написать функцию, которая вычисляет значение выражения $y = x^2 + 3x - 7$ для всех x на отрезке от -5 до 5 с шагом, равным 2.

3.4. Содержание отчета

1. Название работы.
2. Цель работы.
3. Формулировка тренировочного задания и задания по варианту.
4. Файлы с выполненными заданиями 3.1, 3.2.

3.5. Контрольные вопросы

1. Что такое интерактивные скрипты?
2. Как создать интерактивный скрипт?
3. Каковы правила записи команд в интерактивных скриптах?
4. Каким образом вызывается программа, записанная интерактивном скрипте?
5. Какой командой можно ввести данные с клавиатуры?
6. Какая функция позволяет выводить информацию на экран?
7. Какие операторы цикла вы знаете?
8. Какие операции отношения и логические операции вы знаете?
9. С помощью каких операторов осуществляется ветвление алгоритма?
10. Опишите процесс написания и запуска скрипта в Engee**.**

ЗАКЛЮЧЕНИЕ

Настоящее учебно-методическое пособие подводит итог начальному этапу освоения российской платформы математических вычислений и динамического моделирования Engee. В ходе выполнения последовательного цикла практических работ были рассмотрены фундаментальные аспекты работы в среде: от базового интерфейса и вычислений до операций с матрицами и основ программирования, что составляет ядро компетенций современного инженера и исследователя.

Выполнение предложенного комплекса заданий, включающего как подробные инструкции, так и вариативные индивидуальные задачи, позволяет констатировать достижение заявленной цели — формирования у обучающихся не только теоретических знаний, но и практических навыков, необходимых для решения широкого спектра инженерно-технических и научных задач в современной отечественной программной среде. Систематическое освоение материала данного пособия от простого к сложному создает фундамент для дальнейшего профессионального роста и академической мобильности студентов.

Особую значимость имеет тот факт, что приобретенные компетенции не являются узкоспециализированными и замкнутыми исключительно на экосистему платформы Engee. Понимание принципов работы с данными, выполнения матричных операций и основ алгоритмизации, сформированное в ходе обучения, носит универсальный, метапредметный характер. Этот интеллектуальный багаж может быть применен при работе с другими программными средами, фреймворками и языками программирования, что повышает общую ИТ-грамотность и адаптивность будущего специалиста.

Логичной и перспективной целью дальнейшего развития данного учебного курса видится планомерный переход к освоению более сложных и специализированных разделов. В их числе — углубленное изучение возможностей визуального динамического моделирования, работа с графическим редактором для построения сложных систем, освоение продвинутых техник программирования, а также применение всего арсенала средств Engee для решения прикладных задач в таких областях, как теория автоматического управления, цифровая обработка сигналов, машинное обучение и физическое моделирование. Данное пособие станет для студентов базой для их будущих профессиональных и исследовательских достижений в мире высоких технологий и инноваций.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Батенков, К. А. Основы алгоритмизации : учебное пособие / К. А. Батенков, А. В. Козленко. — Москва : РТУ МИРЭА, 2025. — 211 с. — URL: https://e.lanbook.com/book/493370 (дата обращения: 22.01.2026). — Режим доступа: для авториз. пользователей. — ISBN 978-5-7339-2444-1. — Текст : электронный.

2. Документация Engee. – Текст: электронный// Engee.com [сайт]. – URL: https://engee.com/helpcenter/stable/ru/index.html.

3. Мамонова, Т. Е. Информационные технологии. Лабораторный практикум : учебное пособие для вузов / Т. Е. Мамонова. — Москва : Юрайт, 2022. — 176 с. — (Высшее образование). — URL: https://urait.ru/bcode/490340 (дата обращения: 22.01.2026). — ISBN 978-5-9916-7060-9. — Текст: электронный.

4. Бурьков, Д. В. Прикладные программные пакеты для технических специальностей : учебное пособие / Д. В. Бурьков. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2024. — 211 с. — URL: https://www.iprbookshop.ru/146906.html (дата обращения: 23.01.2026). — Режим доступа: для авторизир. пользователей. — ISBN 978-5-9275-4777-7. — Текст : электронный.

5. Курош, А. Г. Курс высшей алгебры : учебник для вузов / А. Г. Курош. — 27-е изд., стер. — Санкт-Петербург : Лань, 2026. — 432 с. — URL: https://e.lanbook.com/book/507517 (дата обращения: 22.01.2026). — Режим доступа: для авториз. пользователей. — ISBN 978-5-507-54342-7. — Текст: электронный.

6. Владимирский, Б. М. Математика. Общий курс : учебник / Б. М. Владимирский, А. Б. Горстко, Я. М. Ерусалимский. — 4-е изд., стер. — Санкт-Петербург : Лань, 2022. — 960 с. — URL: https://e.lanbook.com/book/210206 (дата обращения: 22.01.2026). — Режим доступа: для авториз. пользователей. — Текст: электронный.

7. Шиндин, А. В. Язык программирования математических вычислений JULIA. Базовое руководство: учебно-методическое пособие / А. В. Шиндин. — Нижний Новгород : ННГУ им. Н. И. Лобачевского, 2016. — 24 с. — URL: https://e.lanbook.com/book/153036 (дата обращения: 22.01.2026). — Режим доступа: для авториз. пользователей. — Текст: электронный.

8. Белов Г. В. О возможностях использования языка программирования Julia для решения научных и технических задач / Г. В. Белов, Н. М. Аристова. — Текст : электронный // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. — 2020. — № 2. — С. 27-43. — DOI: 10.18698/0236-3933-

2020-2-27-43. — URL: https://cyberleninka.ru/article/n/o-vozmozhnostyah-ispolzovaniya-yazyka-programmirovaniya-julia-dlya-resheniya-nauchnyh-i-tehnicheskih-zadach (дата обращения: 22.01.2026).

9. Антонюк, В. А. Язык Julia как инструмент исследователя / В. А. Антонюк. – М. : Физический факультет МГУ им. М. В. Ломоносова, 2019. – 48 с. — URL: https://cmp.phys.msu.ru/sites/default/files/VA_Antonyk_Julia_2019.pdf?ysclid=m3747p6z2v59454499 (дата обращения: 22.01.2026). — Текст: электронный.

ПРИЛОЖЕНИЕ А
Примеры выполнения операций с матрицами в Engee
(справочное)

1. Задание матрицы и обращение к ее элементам (знак : (двоеточие) обозначает, что в рассмотрение берутся все элементы)

In [ ]:
```
A = [1 2 3; 4 5 6; 7 8 9]

# Изменение 1-го столбца матрицы

A[1,1] = 100

A
```

Out[0]:
```
3×3 Matrix{Int64}:
 100  2  3
   4  5  6
   7  8  9
```

In [ ]:
```
# Изменение третьего столбца матрицы (поэлементное)

A[:,3] .= 50


# Изменение 2-й строки матрицы

A[2,:] .= 33

A
```

Out[0]:
```
3×3 Matrix{Int64}:
 100   2  50
  33  33  33
   7   8  50
```

2. Вычисление суммы по столбцам единичной матрицы

In [ ]:
```
B = ones(3,3)

B
```

Out[0]:
```
3×3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0
```

```
 1.0  1.0  1.0
```

In [ ]:

```
sum(B)
```

Out[0]:

```
9.0
```

In [ ]:

```
sum(B, dims=1)
```

Out[0]:

```
1×3 Matrix{Float64}:
 3.0  3.0  3.0
```

In [ ]:

```
sum(B, dims=2)
```

Out[0]:

```
3×1 Matrix{Float64}:
 3.0
 3.0
 3.0
```

3. Матричное и поэлементное умножение квадратных матриц

In [ ]:

```
a = [1 2; 3 4]
a
```

Out[0]:

```
2×2 Matrix{Int64}:
 1  2
 3  4
```

In [ ]:

```
b = [3 1; 2 0]
b
```

Out[0]:

```
2×2 Matrix{Int64}:
 3  1
 2  0
```

In [ ]:

```
a*b
```

Out[0]:

```
2×2 Matrix{Int64}:
  7  1
 17  3
```

In [ ]:

```
a .* b
```

Out[0]:

```
2×2 Matrix{Int64}:
 3  2
 6  0
```

4. Возведение матрицы в степень

In [ ]:

```
A = [1 2 3; 0 2 0; 0 0 3]
A
```

Out[0]:

```
3×3 Matrix{Int64}:
 1  2  3
 0  2  0
 0  0  3
```

In [ ]:

```
A^2
```

Out[0]:

```
3×3 Matrix{Int64}:
 1  6  12
 0  4   0
 0  0   9
```

5. Нахождение обратной матрицы

In [ ]:

```
A = [1 2; 0 2]
A
```

Out[0]:

```
2×2 Matrix{Int64}:
 1  2
 0  2
```

In [ ]:

```
B = inv(A)
```

Out[0]:

```
2×2 Matrix{Float64}:
 1.0  -1.0
 0.0   0.5
```

In [ ]:

```
C = A^(-1)
```

Out[0]:

```
2×2 Matrix{Float64}:
 1.0  -1.0
```

```
 0.0   0.5
```

6. Поэлементное прямое и обратное деление векторов

In [ ]:
```
q1 = [1 2 3 4]; q2 = [10 20 30 40];
display(q1)
display(q2)
```

```
1×4 Matrix{Int64}:
 1  2  3  4
1×4 Matrix{Int64}:
 10  20  30  40
```

In [ ]:
```
# Поэлементное прямое деление векторов
p_1 = q1 ./ q2
```

Out[0]:
```
1×4 Matrix{Float64}:
 0.1  0.1  0.1  0.1
```

In [ ]:
```
# Поэлементное обратное деление векторов
p_2 = q1 .\ q2
```

Out[0]:
```
1×4 Matrix{Float64}:
 10.0  10.0  10.0  10.0
```

7. Транспонирование вещественной матрицы B = A^T

In [ ]:
```
A = [1 1 1; 2 2 2; 4 5 6]
```

Out[0]:
```
3×3 Matrix{Int64}:
 1  1  1
 2  2  2
 4  5  6
```

In [ ]:
```
B = A'
```

Out[0]:
```
3×3 adjoint(::Matrix{Int64}) with eltype Int64:
 1  2  4
 1  2  5
 1  2  6
```

In [ ]:

```
B = transpose(A)
```

Out[0]:

```
3×3 transpose(::Matrix{Int64}) with eltype Int64:
 1  2  4
 1  2  5
 1  2  6
```

8. Заданы две матрицы A и B. Необходимо выполнить операцию A^2 + B^2 - (AB)^T:

In [ ]:

```
A = [1 0 1; 2 3 4; -1 6 7];
B = [7 4 2; 3 5 6; -1 2 1];
C = A^2 + B^2 - (A*B)'
```

Out[0]:

```
3×3 Matrix{Int64}:
 53  39  44
 28  51  44
 -1  42  42
```

9. Конкатенация (объединение) матриц по горизонтали

In [ ]:

```
M1 = [1 2; 3 4];
M2 = [5 6 7; 8 9 10];
N = [M1 M2]
```

Out[0]:

```
2×5 Matrix{Int64}:
 1  2  5  6   7
 3  4  8  9  10
```

10. Конкатенация (объединение) матриц с помощью функций cat/vcat/hcat:

cat(<направление>, <матрица_1>, <матрица_2>,…,<матрица_n>, dims=3);
vcat(<направление>, <матрица_1>, <матрица_2>,…,<матрица_n>);

Параметр <направление> может принимать значения:
1 – соответствует объединению по вертикали,
2 – горизонтали,
3 – объединить вдоль третьей оси.

In [ ]:

```
N = hcat( [2; 2], M1, M2 )
```

Out[0]:

```
2×6 Matrix{Int64}:
 2  1  2  5  6   7
 2  3  4  8  9  10
```

In [ ]:

```
N = cat( [2; 2], M1, M2, dims=2 )
```

Out[0]:

```
2×6 Matrix{Int64}:
 2  1  2  5  6   7
 2  3  4  8  9  10
```

11. Объединение матриц по вертикали

In [ ]:

```
M1 = [1 2 3]; M2 = [5 6 7; 8 9 10];
N = [M1; M2]
```

Out[0]:

```
3×3 Matrix{Int64}:
 1  2   3
 5  6   7
 8  9  10
```

In [ ]:

```
cat( M1, M2, dims=1 )
```

Out[0]:

```
3×3 Matrix{Int64}:
 1  2   3
 5  6   7
 8  9  10
```

12. Сумма по строкам

In [ ]:

```
A = [1 2; 3 4]
```

Out[0]:

```
2×2 Matrix{Int64}:
 1  2
 3  4
```

In [ ]:

```
B = sum( A, dims=2 )
```

Out[0]:

```
2×1 Matrix{Int64}:
 3
 7
```

13. Сумма по столбцам

In [ ]:
```
A = [1 2; 3 4]
```

Out[0]:
```
2×2 Matrix{Int64}:
 1  2
 3  4
```

In [ ]:
```
sum(A, dims=1)
```

Out[0]:
```
1×2 Matrix{Int64}:
 4  6
```

14. Сумма всех элементов матрицы

In [ ]:
```
A = [1 2; 3 4]
```

Out[0]:
```
2×2 Matrix{Int64}:
 1  2
 3  4
```

In [ ]:
```
B = sum( A )
```

Out[0]:
```
10
```

15. Выделение главной диагонали матрицы

In [ ]:
```
A = [1 2 3; 1 2 3; 1 2 3 ]
```

Out[0]:
```
3×3 Matrix{Int64}:
 1  2  3
 1  2  3
 1  2  3
```

In [ ]:
```
using LinearAlgebra
B = diag( A )
```

Out[0]:
```
3-element Vector{Int64}:
 1
 2
 3
```

16. Выделение побочной диагонали, расположенной ниже главной

In [ ]:
```
A = [1 2 3; 1 2 3; 1 2 3]
```

Out[0]:
```
3×3 Matrix{Int64}:
 1  2  3
 1  2  3
 1  2  3
```

In [ ]:
```
B = diag(A, -1)
```

Out[0]:
```
2-element Vector{Int64}:
 1
 2
```

17. Выделение побочной диагонали, расположенной выше главной

In [ ]:
```
A = [1 2 3; 1 2 3; 1 2 3]
```

Out[0]:
```
3×3 Matrix{Int64}:
 1  2  3
 1  2  3
 1  2  3
```

In [ ]:
```
B = diag(A, 1)
```

Out[0]:
```
2-element Vector{Int64}:
 2
 3
```

18. Построение матрицы на основе заданной диагонали: элементы заданного *вектора* будут располагаться на главной диагонали созданной матрицы

In [ ]:
```
A = [1, 2, 3]
```

Out[0]:
```
3-element Vector{Int64}:
 1
 2
 3
```

In [ ]:
```
B = Diagonal(A)
```

Out[0]:

```
3×3 Diagonal{Int64, Vector{Int64}}:
 1  ·  ·
 ·  2  ·
 ·  ·  3
```

19. Элементы заданного вектора будут располагаться ниже главной диагонали созданной матрицы

In [ ]:

```
using LinearAlgebra
A = [1 2 3]
B = diagm( -1 => A[2:3] )
```

Out[0]:

```
3×3 Matrix{Int64}:
 0  0  0
 2  0  0
 0  3  0
```

20. Элементы заданного вектора будут располагаться выше главной диагонали созданной матрицы

In [ ]:

```
A = [1 2 3]
B = diagm( 1 => A[2:3] )
```

Out[0]:

```
3×3 Matrix{Int64}:
 0  2  0
 0  0  3
 0  0  0
```

21. Подсчет количества строк и количества столбцов объекта

In [ ]:

```
A = [1; 2; 3; 4];
B = [1 2 3 4];
display( size(A) )
display( size(B) )
```

```
(4,)
(1, 4)
```

22. Общее количество элементов массива

In [ ]:

```
A = [1; 2; 3; 4]; B = [1 2 3 4];
display( length(A) )
display( length(B) )
```

```
4
4
```

23. Количество элементов вектора или строки матрицы

In [ ]:

```
M = [1 2 3; 4 5 6]
```

Out[0]:

```
2×3 Matrix{Int64}:
 1  2  3
 4  5  6
```

In [ ]:

```
length(M)
```

Out[0]:

```
6
```

24. Сравнение команд size и length

In [ ]:

```
A = rand(5,5)
```

Out[0]:

```
5×5 Matrix{Float64}:
 0.908502  0.545855  0.532986  0.539892  0.18507
 0.632128  0.927774  0.819572  0.281962  0.489637
 0.217912  0.325548  0.668384  0.591432  0.893202
 0.67997   0.163777  0.88375   0.211613  0.00858644
 0.612217  0.933909  0.498055  0.242342  0.987897
```

In [ ]:

```
size(A)
```

Out[0]:

```
(5, 5)
```

In [ ]:

```
length(A)
```

Out[0]:

```
25
```

ПРИЛОЖЕНИЕ Б
Основные математические операции над матрицами
(справочное)

1. Сложение матриц

Определение.

Суммой матриц *A и B одного размера называется матрица C = A + B такого же размера, получаемая из исходных путем сложения соответствующих элементов.*

Операция сложения определена для матриц одного размера.
Пусть

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix}$$

Суммой матриц *A* и *B* называется матрица

$$A + B = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{pmatrix}$$

Пример 1.
Сложим две матрицы

$$A = \begin{pmatrix} 0 & -1 & -5 \\ 4 & 2 & 3 \end{pmatrix} \text{ и } B = \begin{pmatrix} 6 & 3 & 0 \\ 1 & 4 & -2 \end{pmatrix}$$

$$C = A + B = \begin{pmatrix} 0+6 & -1+3 & -5+0 \\ 4+1 & 2+4 & 3-2 \end{pmatrix} = \begin{pmatrix} 6 & 2 & -5 \\ 5 & 6 & 1 \end{pmatrix}$$

Определение.

Матрица B называется ***противоположной*** *матрице A и обозначается – A, если A + B = 0.*

2. Умножение матрицы на число

Определение.

Произведением матрицы на число *называется матрица, полученная из исходной умножением каждого ее элемента на заданное число.*

$$\lambda A = \begin{pmatrix} \lambda a_{11} & \lambda a_{12} & \cdots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \cdots & \lambda a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda a_{m1} & \lambda a_{m2} & \cdots & \lambda a_{mn} \end{pmatrix}$$

Пример 2.

Умножим матрицу $A = \begin{pmatrix} 1 & 0 \\ 6 & -6 \\ 12 & 3 \end{pmatrix}$ на действительное число $\lambda = \frac{2}{3}$

$$B = \lambda A = \begin{pmatrix} \frac{2}{3}\cdot 1 & \frac{2}{3}\cdot 0 \\ \frac{2}{3}\cdot 6 & \frac{2}{3}\cdot(-6) \\ \frac{2}{3}\cdot 12 & \frac{2}{3}\cdot 3 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & 0 \\ 4 & -4 \\ 8 & 2 \end{pmatrix}$$

3. Умножение матриц

Определение

Произведением матрицы *A mxn на матрицу B nxk называется матрица C mxk такая, что элемент матрицы C, стоящий в i-й строке и j-м столбце, т.е. элемент Cij, равен сумме произведений элементов i-й строки матрицы A на соответствующие элементы j-го столбца матрицы B.*

Матрицу *A* можно умножить на матрицу *B* только в том случае, если число столбцов матрицы *A* совпадает с числом строк матрицы *B*.

Пусть

$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}$ и $B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$

Произведением матриц A и B будет матрица C, элементы которой находятся следующим образом

$$C = A \cdot B = AB = \begin{pmatrix} a_{11}\cdot b_{11} + a_{12}\cdot b_{21} & a_{11}\cdot b_{12} + a_{12}\cdot b_{22} & a_{11}\cdot b_{13} + a_{12}\cdot b_{23} \\ a_{21}\cdot b_{11} + a_{22}\cdot b_{21} & a_{21}\cdot b_{12} + a_{22}\cdot b_{22} & a_{21}\cdot b_{13} + a_{22}\cdot b_{23} \\ a_{31}\cdot b_{11} + a_{32}\cdot b_{21} & a_{31}\cdot b_{12} + a_{32}\cdot b_{22} & a_{31}\cdot b_{13} + a_{32}\cdot b_{23} \end{pmatrix}$$

Пример 3.

Перемножим две матрицы

$A = \begin{pmatrix} 1 & -2 & 3 \\ 4 & 1 & 7 \end{pmatrix}$ и $B = \begin{pmatrix} -9 & 1 & 0 \\ 4 & 1 & 1 \\ -2 & 2 & -1 \end{pmatrix}$

Решение.

Т.к. число столбцов матрицы A совпадает с числом строк матрицы B, то можно говорить о произведении матрицы A на матрицу B:

$$AB = \begin{pmatrix} 1\cdot(-9) + (-2)\cdot 4 + 3\cdot(-2) & 1\cdot 1 + (-2)\cdot 1 + 3\cdot 2 & 1\cdot 0 + (-2)\cdot 1 + 3\cdot(- \\ 4\cdot(-9) + 1\cdot 4 + 7\cdot(-2) & 4\cdot 1 + 1\cdot 1 + 7\cdot 2 & 4\cdot 0 + 1\cdot 1 + 7\cdot(-1 \end{pmatrix}$$

=

$$= \begin{pmatrix} -23 & 5 & -5 \\ -46 & 19 & -6 \end{pmatrix}$$

4. Транспонирование матриц

Определение.

Транспонирование матрицы *- это операция над матрицей, когда ее строки становятся столбцами с теми же номерами.*

При транспонировании строки и столбцы матрицы меняются местами.

Пример 4.

Пусть

$$A = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \quad B = (2 \quad 3 \quad 1 \quad 2), \quad C = \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \quad D = \begin{pmatrix} -1 & 4 \\ 4 & -1 \end{pmatrix}$$

Транспонированными к ним будут матрицы

$$A^T = \begin{pmatrix} 2 & 6 & 1 \\ 3 & 7 & 2 \\ 4 & 8 & 3 \\ 5 & 9 & 4 \end{pmatrix}, \quad B^T = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 2 \end{pmatrix}, \quad C^T = (1 \quad 4), \quad D^T = \begin{pmatrix} -1 & 4 \\ 4 & -1 \end{pmatrix}$$

Замечание. Дважды транспонированная матрица совпадает с первоначальной. $(A^T)^T = A$

Определение. Если $A^T = A$, то матрица A называется **симметричной.**

5. Определитель матрицы и его вычисление

Определение.

Квадратной матрице $A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$ *n-го порядка ставиться в соответствие число* $|A| = det\ A = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}$, *называемое* ***определителем матрицы*** *или* ***детерминантом.***

5.1. Вычисление определителя 2-го порядка

Чтобы ***вычислить определитель матрицы второго порядка****, надо из произведения элементов главной диагонали вычесть произведение элементов побочной диагонали:*

$$|A| = det\ A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

Пример 5.1.

Найдем определитель матрицы $A = \begin{pmatrix} 1 & -6 \\ -3 & 4 \end{pmatrix}$.

$$det\ A = \begin{vmatrix} 1 & -6 \\ -3 & 4 \end{vmatrix} = 1 \cdot 4 - (-6) \cdot (-3) = 4 - 18 = -14$$

5.2. Вычисление определителя n-го порядка

Определение.

Алгебраическим дополнением A_{ij} *элемента* a_{ij} *называется определитель* $(n-1)$ *порядка, полученный из* A *вычеркиванием* i*-й строки и* j*-го столбца, на пересечении которых стоит элемент* a_{ij}*, взятый со знаком* $(-1)^{i+j}$.

Например.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \Rightarrow A_{12} = (-1)^{1+2} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}$$

Определение.

Определитель $|A|$ n *- го порядка равен* $|A| = a_{11} \cdot A_{11} + a_{12} \cdot A_{12} + \ldots + a_{1n} \cdot A_{1n}$.

Такое разложение называется разложением по элементам первой строки. Оно сводит вычисление n *- го порядка к вычислению определителей* $(n-1)$ *- го порядка и т.д., до определителей 2-го порядка.*

Можно доказать, что определитель можно вычислить путем разложения по любой строке или столбцу.

Пример 5.2.

Найдем определитель матрицы $A = \begin{pmatrix} 1 & 2 & 3 \\ -1 & 4 & 2 \\ 0 & 2 & 1 \end{pmatrix}$ путем разложения по первой строке.

$$det\ A = \begin{vmatrix} 1 & 2 & 3 \\ -1 & 4 & 2 \\ 0 & 2 & 1 \end{vmatrix} = 1 \cdot (-1)^{1+1} \begin{vmatrix} 4 & 2 \\ 2 & 1 \end{vmatrix} + 2 \cdot (-1)^{1+2} \begin{vmatrix} -1 & 2 \\ 0 & 1 \end{vmatrix} + 3 \cdot (-1)^{1+3} \begin{vmatrix} -1 & 4 \\ 0 & 2 \end{vmatrix} = -4$$

.

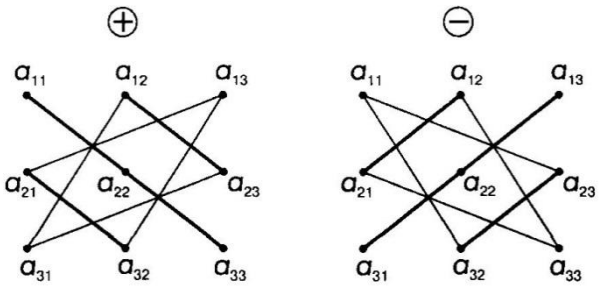
Правило треугольника для определителя 3-го порядка

Определение.

Определителем третьего порядка квадратной матрицы третьего порядка называется число, вычисляемое по формуле:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \cdot a_{22} \cdot a_{33} + a_{12} \cdot a_{23} \cdot a_{31} + a_{13} \cdot a_{21} \cdot a_{32} -$$

$$-a_{13} \cdot a_{22} \cdot a_{31} - a_{11} \cdot a_{23} \cdot a_{32} - a_{12} \cdot a_{21} \cdot a_{33}$$



Пример 5.3.

Найдем определитель 3-го порядка матрицы $A = \begin{pmatrix} -2 & 3 & -4 \\ 5 & 1 & 2 \\ -3 & 4 & -5 \end{pmatrix}$, применяя правило треугольника.

$$det\ A = \begin{vmatrix} -2 & 3 & -4 \\ 5 & 1 & 2 \\ -3 & 4 & -5 \end{vmatrix} = -2 \cdot 1 \cdot (-5) + 5 \cdot 4 \cdot (-4) + 3 \cdot 2 \cdot (-3) - (-3) \cdot 1 \cdot (-4) -$$

$4 \cdot 2 \cdot (-2) - 5 \cdot 3 \cdot (-5) = = 10 - 80 - 18 - 12 + 16 + 75 = -9.$

6. Обратная матрица и ее вычисление

Определение.

*Матрица A^{-1} называется **обратной** к матрице A, если при умножении ее на матрицу A, как справа, так и слева, получится единичная матрица.*

$$A^{-1} \times A = A \times A^{-1} = E$$

***Единичная матрица** – квадратная матрица, элементы главной диагонали которой равны единице поля, а остальные равны нулю.*

$$E_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Матрица называется ***невырожденной***, если ее определитель не равен 0, и называется *вырожденной*, если ее определитель равен 0.

Обратная матрица A^{-1} существует только тогда, когда матрица невырожденная, т.е. $|A| \neq 0$.

Алгоритм нахождения:

1. Найти определитель матрицы A.

Если $|A| = 0$, то обратная матрица не существует, если $|A| \neq 0$, то перейти ко второму шагу.

2. Найти матрицу A^T, транспонированную к матрице A.

3. Найти алгебраические дополнения элементов матрицы A^T и составить из них матрицу $\tilde{A}$, которая называется *присоединенной*.

$$\tilde{A} = \begin{pmatrix} A^T_{11} & \ldots. & A^T_{1n} \\ A^T_{21} & \ldots. & A^T_{2n} \\ A^T_{m1} & \ldots. & A^T_{mn} \end{pmatrix}$$

4. Обратную матрицу найти по формуле:

$$A^{-1} = \frac{1}{|A|} \times \tilde{A}$$

5. Сделать проверку $A^{-1} \times A = E$

Пример 6.

Найдем обратную матрицу к матрице $A = \begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix}$.

1. Находим определитель матрицы A:

$|A| = \begin{vmatrix} 3 & 2 \\ 1 & 1 \end{vmatrix} = 3 - 2 = 1 \neq 0$. Он не равен нулю, значит, обратная матрица существует.

2. Находим матрицу A^T:

$$A^T = \begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix}.$$

3. Находим алгебраические дополнения элементов матрицы A^T и составляем из них присоединенную матрицу $\tilde{A}$:

$$A^T_{11} = (-1)^2 \cdot 1 = 1;$$
$$A^T_{12} = (-1)^{1+2} \cdot 2 = -2;$$
$$A^T_{21} = (-1)^{2+1} \cdot 1 = -1;$$
$$A^T_{22} = (-1)^{2+2} \cdot 3 = 3;$$
$$\tilde{A} = \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}.$$

4. Обратную матрицу находим по формуле

$$A^{-1} = \frac{1}{|A|} \times \tilde{A} = \frac{1}{1} \times \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}.$$

5. Сделаем проверку $A^{-1} \times A = E$

$$\begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix} \times \begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot 3 + (-2) \cdot 2 & 1 \cdot 2 + (-2) \cdot 1 \\ -1 \cdot 3 + 3 \cdot 1 & -1 \cdot 2 + 3 \cdot 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = E$$

У ч е б н о е и з д а н и е

Мирянова Вера Николаевна
Мирянова Анна Дмитриевна

ПРАКТИКУМ В ENGEE. ОСНОВЫ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие

В авторской редакции

Изд. № 23/2026. Объем 5 п. л. Усл. печ. л. 4,65. Уч.-изд. л. 4,9.
Издательский центр ФГАОУ ВО «Севастопольский государственный университет»