БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ
И ИНФОРМАТИКИ
Кафедра компьютерных технологий и систем

В. Б. Таранчук

ИНСТРУМЕНТЫ ИНТЕРАКТИВНОГО ПРОГРАММИРОВАНИЯ В СИСТЕМЕ *MATHEMATICA*

Учебные материалы
для студентов факультета
прикладной математики и информатики

МИНСК
2022

УДК 519.67(075.8)+004.42(075.8)
ББК 22.19я73-1+32.973-018я73-1
Т19

Рекомендовано советом
факультета прикладной математики и информатики БГУ
22 ноября 2022 г., протокол № 4

Р е ц е н з е н т
доктор физико-математических наук *Н. Н. Гринчик*

Таранчук, В. Б.

Т19 Инструменты интерактивного программирования в системе *Mathematica* : учеб. материалы для студентов фак. прикладной математики и информатики / В. Б. Таранчук. – Минск : БГУ, 2022. –50 с.

Изложены многочисленные варианты интерактивной работы в системе *Mathematica* с выражениями, примеры использования разных форм записи функций системы, в частности, анонимных, особенности их применения поясняются в задачах в 2D, 3D визуализации. Интерактивное программирование на языке Wolfram Language с использованием функций формирования, редактирования списков иллюстрируется решениями упражнений и задач для избранных на серии цикла «Список целых чисел».

Предназначено для студентов факультета прикладной математики и информатики.

УДК 519.67(075.8)+004.42(075.8)
ББК 22.19я73-1+32.973-018я73-1

ПРЕДИСЛОВИЕ

В учебных материалах изложены теоретические и методические вопросы, примеры интерактивного решения возникающих в практике вычислительных экспериментов, интеллектуального анализа данных задач с использованием языка Wolfram Language, системы компьютерной алгебры Wolfram *Mathematica*. Теоретические и методические вопросы интерактивного программирования изучаются в дисциплинах специализации «Когнитивная визуализация» (учебная дисциплина для специальности 1-31 03 03 Прикладная математика, направление специальности 1-31 03 03-01 Прикладная математика, научно-производственная деятельность), «Компьютерный анализ и визуализация» (учебная дисциплина для специальности 1-31 03 07 Прикладная информатика, направление специальности 1-31 03 07-01 Прикладная информатика, программное обеспечение компьютерных систем), «Технологии интерактивной визуализации» (учебная дисциплина для специальности 1-31 03 04 Информатика), «Интерактивные вычисления и визуализация» (учебная дисциплина для специальности 1-31 03 03 Прикладная математика, направление специальности 1-31 03 03-01 Прикладная математика, научно-производственная деятельность).

В изложенных материалах лекций и заданиях практических занятий, в процессе преподавания акценты сделаны на:

- освоение обучаемыми методов функционального программирования и инструментов создания в Wolfram *Mathematica* интерактивных программных модулей с возможностями интеллектуальной обработки данных; средств манипулирования функциями и данными;
- приобретение студентами навыков программирования и отладки блокнотов системы *Mathematica*, выполнения символьных вычислений, обработки данных разной структуры и содержания;
- контроль знаний и навыков средствами компьютерного тестирования (на каждой лекции – 2-5 контрольных вопросов или кратких заданий, на каждом практическом занятии – 5-6 вопросов и 7-9 заданий функционального программирования).

Советы и критические замечания по изданию просьба направлять на taranchuk@bsu.by.

Инструменты интерактивного программирования в системе *Mathematica*

Таранчук Валерий Борисович
БГУ,
факультет прикладной математики и информатики

Учебные материалы, инструкции и рекомендации пользователям системы компьютерной алгебры *Mathematica*, обучающие примеры и упражнения

Материалы темы “Wolfram Language. Выражения. Функции”

- ✓ Базовые понятия, основные возможности системы *Mathematica*
- ✓ Траектория, знаковые пункты истории системы *Mathematica*
- ✓ Выражения. Работа с выражениями
- ✓ Части выражений, удаление элементов
- ✓ Манипуляции с выражениями.
- ✓ Функции. Формы записи.
- ✓ Анонимные функции.
- ✓ Примеры использования анонимных функций в 2D, 3D визуализации
- ✓ Тестовые задания для самоконтроля

Базовые понятия, основные возможности системы *Mathematica*

Работа с материалами данной темы предполагает, что изучены и освоены основы системы Wolfram *Mathematica*. Тем не менее, напомним несколько принципиальных позиций.

Mathematica представляет собой модульную систему программного обеспечения, где ядро, которое фактически выполняет вычисления, отделено от интерфейса (оболочки), отвечающего за взаимодействие с пользователем. Самым распространенным способом работы с *Mathematica* является использование интерактивных документов, известных как “блокноты” [1 – 3]. Блокноты позволяют “смешивать” в любой пропорции исходные данные и результаты вычислений *Mathematica* с текстом, графикой, формулами и другими материалами. Можно использовать блокнот, как для выполнения текущих расчетов, так и в качестве средства для презентации или публикации результатов работы. В блокнотах находятся, как код, так и результаты вычислений, аналитические преобразования, вычисления. Работа с документами в системе сводится к набору в секциях ввода выражений и их исполнению. В интерфейсе (оболочке) “блокнота” пользователь взаимодействует с *Mathematica* путем интерактивного создания документов. Интерфейс системы содержит много меню и графических инструментов для создания, редактирования и просмотра документов, для отправки данных к ядру и получения результатов от ядра. Все вычисления в *Mathematica* выполняются в блокнотах, которые имеют расширение .nb – файлы текстового формата. Эти файлы могут редактироваться любым текстовым редактором, поддерживающим формат ASCII.

Блокноты могут быть разбиты на секции (ячейки) различного типа:

✓ Секции (ячейки) ввода – в них задаются команды (функции), которые будут вычислены.

✓ Секции (ячейки) результата – в них выводятся результаты вычислений.

✓ Другие секции – ячейки с текстом, заголовки, предупреждения, ...

Для получения результата достаточно, находясь в исполняемой секции (секции ввода), нажать одновременно клавиши SHIFT и ENTER. Другие способы выполнения секций подробно описаны в [2, 3].

Помощь и справка в *Mathematica*.

Центр документации (Documentation Center), Навигатор по функциям (Function Navigator), Виртуальный учебник (Virtual Book) обеспечивают помощь пользователям в изучении программного языка и функциональных возможностей системы *Mathematica*. Встроенная документация системы содержит свыше 150 тысяч представительных и наглядных примеров кодов на языке Wolfram. Все документы полностью интерактивны, это документы *Mathematica*, в которых можно пробовать свои коды, модифицировать готовые примеры непосредственно в справке.

Правила и приемы использования встроенного Центра документации, Навигатора по функциям, Виртуального учебника, важные сочетания клавиш; как выполнять поиск в системе Справки и Помощи подробно описаны в [2], примерами проиллюстрированы возможности интерфейса пользователя системы, приемы использования и вариантов действий пользователя с сервисами: Контекстный модуль помощи ввода, Предсказательный интерфейс, Линейка предложений о следующих вычислениях.

Синтаксис языка *Mathematica* построен с использованием единого принципа названия встроенных функций (их число превышает 6000): каждой функции присвоено имя, начинающееся с заглавной буквы; каждое слово внутри имени также пишется с заглавной буквы (так называемый CamelCase – стиль написания составных слов, при котором несколько слов пишутся слитно без пробелов, и каждое слово пишется с заглавной буквы); не применяются аббревиатуры (только полные английские слова) [1, 4]. Все встроенные функции *Mathematica* называются в соответствии с тем, что они делают. Имена встроенных функций могут иметь большую длину, например:

- InverseLaplaceTransform – обратное преобразование Лапласа,
- DiscreteWaveletPacketTransform – дискретное пакетное вейвлет преобразование,
- ListVectorDensityPlot – плотностно-векторная диаграмма по данным,
- MultivariateHypergeometricDistribution – многопараметрическое гипергеометрическое распределение.

Траектория, знаковые пункты истории системы *Mathematica*

Много лет разработчики называли свою основную систему *Mathematica*. Однако после того, как были добавлены новые направления, в которых инструменты системы вышли далеко за пределы вещей, связанных на прямую с «математикой», принято решение, начиная с 11-ой версии, ввести понятие Wolfram Language, чтобы передать суть всего того, что можно и делается. С упомянутого этапа Wolfram Language определяет работу не только системы *Mathematica*, но и Wolfram Development Platform, Wolfram Programming Lab, а также других продуктов и платформ.

Возможности Wolfram Mathematica версии 8 изложены в [5], полные списки обновлений в следующих версиях можно просмотреть на страницах [6 – 10]. Кратко обозначим знаковые решения, функции, которые изучаются в рамках отмеченных выше дисциплин специализаций.

Начиная с версии 9:

В системе *Mathematica* 9 добавлены новые крупные составляющие – ещё более расширяя не имеющую себе равных базу алгоритмических, информационных возможностей и возможностей пользовательского интерфейса системы [6].

Линейка предложений о следующих вычислениях. Как только закончены вычисления, вам предлагаются оптимизированные предложения о следующих возможных шагах. Нажатием кнопки можно выполнить новую функцию или открыть модуль оперативной помощи. Этот новый подход к интерфейсу пользователя позволяет ориентироваться в функциях системы *Mathematica* и открывать для себя новые функциональные возможности.

Развёрнутая поддержка теории вероятностей и статистики. Расширенные возможности по теории вероятностей и статистике, в том числе, критерии статистической независимости, новые тесты для проверки статистических гипотез, поддержка взвешенных данных, новые параметрические и вторичные распределения вероятностей.

Обработка 3D объемных изображений. Подсистема по обработке изображений поддерживает операции с трехмерными объемными изображениями, такие как пиксельные операции, локальное фильтрование, и морфологические операции; рендеринг трехмерных поверхностей и объёмов.

Значительные расширения в нахождении численных решений дифференциальных уравнений. Решение дифференциальных уравнений с разрывными коэффициентами, гибридных дискретно-непрерывных динамических систем, параметрических дифференциальных уравнений, дифференциально-алгебраических уравнений.

Расширенные визуализация и элементы управления. Настраиваемые интерактивные индикаторы для приборных досок и элементов управления, системная поддержка автоматических легенд на графиках и диаграммах, новые функции визуализации, специализированные для обработки сигналов.

Поддержка полного спектра интернет доступа. Полный доступ к интернету со стороны клиента для обмена информацией с удалёнными серверами и для работы с программными интерфейсами веб-приложений.

Начиная с версии 10:

Система *Mathematica* отметила своё 25-летие в 2013 году. Эффективно используя свои основные принципы автоматизации и унификации, она продолжает развиваться с постоянно растущим темпом [7]. *Mathematica* 10 открывает большое количество новых предметных областей, таких как машинное обучение, вычислительная геометрия, географические вычисления и работа с внешними приборами, а также углубляет свои возможности по всему миру алгоритмов. *Mathematica* 10 является первой версией системы, использующей полный язык Wolfram Language.

Решение дифференциальных уравнений. Уравнения в частных производных и конечные элементы. Символьные решения дифференциальных уравнений с запаздыванием. Гибридные дифференциальные уравнения.

Машинное обучение. Машинное обучение с высокой степенью автоматизации. Встроенный комплект классификаторов. Автоматический анализ временных рядов.

Географические вычисления. Географические визуализации. Свойства, связанные с геоположением. Георасчёты с использованием логических объектов.

Машинное обучение. Нейронные сети. Машинное обучение.

Визуализация и графика. Визуализация: аннотации, масштабы, исключения. Новые предметные области визуализации (от анатомии до аудио и автоматов; визуализация данных и времени расширена и включает хронологии для того, чтобы отобразить, когда что произошло, гистограммы для наглядности частоты событий; распределение географических положений может демонстрироваться в качестве гистограмм, на географических картах, которые можно панорамировать, увеличивать и уменьшать; новые функции для визуализации языка и текста облегчают визуализацию тем, используя облака слов, и отображение грамматической структуры предложений). Визуализация объёма.

Расширенный инфобанк Wolfram Knowledgebase. Система *Mathematica* использует в полном объёме недавно пополненный информационный банк Wolfram Knowledgebase через язык Wolfram Language.

Интеграция с Wolfram Cloud. *Mathematica* 10 подключает систему к новой облачной экосистеме Wolfram Cloud.

Начиная с версии 11:

Представляя собой главную веху в беспрецедентном 30-летнем путешествии, версия 11 значительно расширяет охват системы *Mathematica* и внедряет много новшеств, которые обеспечивают пользователям новый уровень мощности и эффективности [8]. Ставя перед собой цель в конечном

итоге интегрировать и автоматизировать все области вычислений и знаний, каждая новая версия *Mathematica* предоставляет доступ к новым предметным областям, как обычный компонент рабочего процесса. *Mathematica* 11 предоставляет интегрированные инструменты, расширяющие охват межпредметных проектов, которые могут повседневно выполняться пользователями на всех уровнях. Система версии 11 основана на недавнем научно-исследовательском прорыве в группе областей, включая вычисление нейронных сетей, аудио интеграцию и вычислительную обработку лингвистических данных. Существующие пользователи системы *Mathematica* также получают широкий спектр удобных улучшений в интерфейсе, языке и глубине и эффективности алгоритмов.

Начиная с версии 12:

Представляя собой еще одну важную веху в уникальном путешествии, охватывающем более 30 лет, значительно расширяет охват системы *Mathematica* и внедряет много новшеств, которые обеспечат всем пользователям Mathematica новый уровень мощности и эффективности [9].

Визуализация и графика. Базовая визуализация (расширена функциональность существующих визуализаций, чтобы упростить извлечение информации из графиков и диаграмм; можно показать экстремальные диапазоны с помощью новой функции масштабирования, поддерживающей большее количество типов графиков; можно вывести полный диапазон неопределенных значений в данных с гибкими полосами ошибок, диапазонами и областями; можно просматривать свои данные по-новому с помощью новых макетов для графиков, которые сопоставляют наборы данных таким образом, чтобы было легко сравнивать значения и тенденции). Визуализация с этикеткой. Комплексная визуализация. Визуализация географических данных.

Машинное обучение. Суперфункции машинного обучения. Фреймворк нейронной сети. Машинное обучение для изображений. Машинное обучение для аудио. Обработка естественного языка.

Wolfram Knowledgebase. Язык запросов к базе знаний. Объекты астрономии и науки о космосе. Биологические и медицинские объекты. Математические объекты. Географические объекты. Финансовые и социально-экономические объекты.

Начиная с версии 13:

Опираясь на более чем треть века непрерывных инноваций, версия 13 как никогда ранее расширяет границы современных вычислений [10]. Внедрение вычислительной парадигмы – язык Wolfram Language предоставляет доступ к вычислительным возможностям на значительно более высоком уровне, чем когда-либо прежде, используя встроенный вычислительный интеллект, который опирается на огромную глубину алгоритмов и реальных знаний, тщательно интегрированных в течение трех

десятилетий. Являясь эффективным для создания, как малых, так и больших программ, которые естественным образом допускают введение в широкое использование, как локально, так и облаке, язык Wolfram Language, основываясь на чётких принципах (и элегантной унифицированной символической структуре) создаёт то, что становится самым продуктивным языком программирования в мире и единственным полномасштабным вычислительным языком, а также первым настоящим вычислительным языком общения между людьми и системами искусственного интеллекта.

Визуализация и графика. Векторная и комплексная визуализация. Многопанельная и многоосевая визуализация. Графическое освещение, наполнители и шейдеры. Новая графика и визуализация.

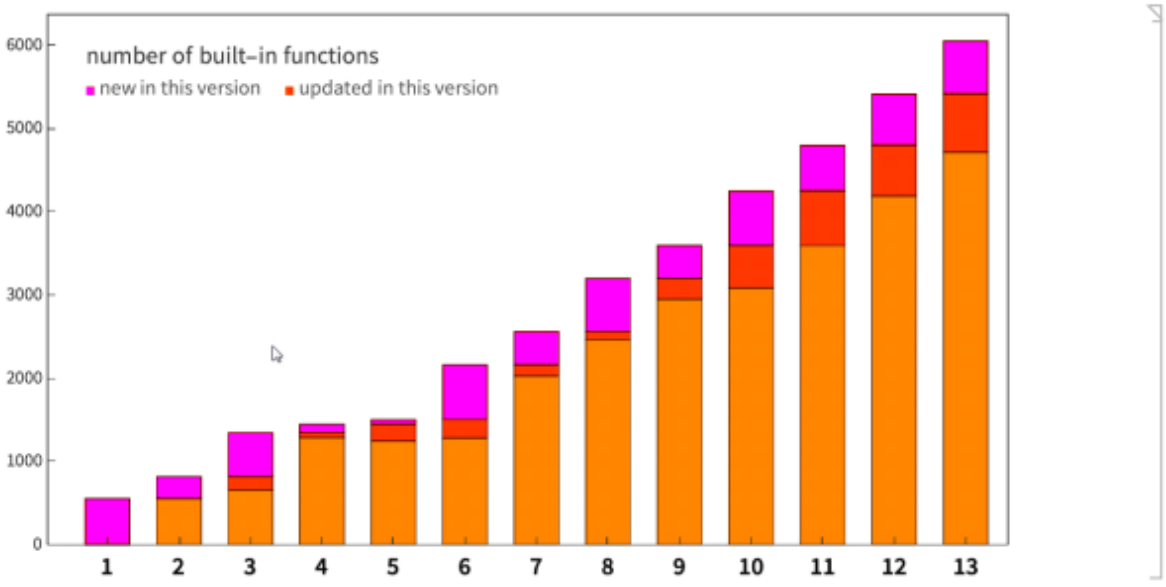
Оптимизация, дифференциальное уравнение в частных производных(PDE) и системное моделирование. Математическая оптимизация. PDE моделирование.

Данные и наука о данных. Машинное обучение и нейронные сети. База знаний. Дата и время. Пространственная статистика.

Блокноты, облако и репозитории. Интерфейс блокнота. Облако и создание веб-страниц. Данные и репозитории функций.

Повторно отметим, что выше перечислены не все нововведения в версиях, а только используемые в рамках изучаемых дисциплин.

График ниже иллюстрирует рост числа функций ядра в версиях системы, сколько вводили новых функций, сколько модернизировали.



Преимущества языка Wolfram Language.

► Язык проблемно ориентированный. Можно программировать "как думаешь", в парадигме естественной для каждой задачи.

► Поддержка современных парадигм программирования с возможностью эмуляции новых для языка парадигм. Основные парадигмы:

функциональное программирование (как LISP, Erlang, APL, ML, F#, Scala, Miranda, Haskell), процедурное программирование (как в Algol, C, COBOL, Fortran, Pascal, PureBasic, PL/I, MATLAB), логическое программирование (как в Planner, Prolog, Mercury, Oz, Visual Prolog, QLISP, Fril), программирование на массивах и структурных операциях (как в APL, MATLAB). Далее на примере вычисления факториала показаны соответствующие варианты программных решений.

► Обширный набор встроенных функций (более 6000), которые охватывают практически все значимые области знания.

► Мощные символьные и численные вычисления + гибридные вычисления; встроенная возможность распараллеливания операций, работы с CUDA, OpenCL, Hadoop.

► Краткий (компактный) код, высокая экспрессивность (код – несколько строчек, а за ними кроется очень большой объём манипуляций); например, коэффициент сопоставления числа символов кода с Java на простейших задачах анализа содержимого массивов – от 7 до 15.

► Возможность решать задачу / разрабатывать код на высоком уровне абстракции без заботы о том, как связаны между собой компоненты решения.

► Высокая степень интеграции компонент, интеграция с другими языками программирования (C, Java, R, MATLAB); возможность использовать совместно функции из разных областей знаний для решения одной сложной задачи.

► Масштабируемость создаваемых программ и приложений.

► Возможности использования и развёртывания языка – на рабочем компьютере, в облаке, мобильных и встраиваемых платформах и ..

► Быстрый цикл разработки, как отдельных программ, так и интерактивных приложений и интерфейсов.

► Высокая степень интерактивности и динамической визуализации результатов (интерактивная разработка в Mathematica FrontEnd, подсветка кода, моментальная визуализация результатов).

► Большое количество форматов импорта/экспорта (172 – импорта, 144 – экспорта) и встроенных парсеров различных данных (около 600), курируемых баз данных (около 90).

► Интеграция с большим количеством онлайн ресурсов (Facebook, Twitter, Instagram, Google+ и др.), сервис интеграции с Википедией, возможность создавать свои методы API и использовать сторонние, которые не встроены в язык напрямую.

► Большое количество фактической информации из реального мира. К примеру, просто запросить данные о планетах (PlanetData), географические (Country, City, AdministrativeDivision, ZIPCode, TimeZone и др.), о известных людях (PersonData), исторические, финансовые и множество других данных.

► Интерактивные документы, с которыми можно работать в

естественном индивидуальном стиле.

► Возможности разработки больших программных приложений. Пакеты расширения (Mathematica packages), профессиональная среда разработки Wolfram Workbench на базе Eclipse, средства разработки (profiler, debugger, unit-testing).

Выражения. Основные формы выражений

В основном, работа в системе *Mathematica* – это операции с объектами, которыми являются математические выражения (expr), символы (symbols), строки из символов (strings), числа различного типа, константы, переменные, списки, мультимедиа объекты и т.д. Каждый используемый в системе объект характеризуется своим именем – идентификатором, которое должно быть уникальным. В *Mathematica* приняты следующие правила задания имен:

name – имя объекта, заданного пользователем;

Name – имя объекта, входящего в ядро системы;

$Name – имя системного объекта.

Все объекты (например, функции), включенные в ядро, имеют имя, начинающееся с большой буквы (например, Rationalize, Simplify, Sin, Plot). Относящиеся к системе объекты (в частности, символы, глобальные переменные и функции) начинаются со знака $ – например, $Aborted, $FrontEnd, $MaxPrecision, $ProcessorCount, $ProcessorType. Заданные пользователем объекты следует именовать прописными (малыми) буквами на первом месте и любыми на следующих. Объекты (чаще всего это функции), встроенные в систему, принято называть внутренними или встроенными. Объекты, которые создает пользователь (в том числе, используя внутренние объекты), называют внешними объектами. К ним, в частности, относятся процедуры и функции, составляемые пользователем.

Одним из важнейших понятий системы *Mathematica* является математическое выражение, или просто выражение (от английского слова expression). Основой для построения выражений являются атомарные выражения: символы, числа, строки. Особое место отводится спискам. Работа с математическими выражениями в символьном виде, умение их преобразовывать и выделять в них нужные фрагменты – основа основ символьной математики.

В *Mathematica* всё является выражениями, которые состоят из символов и скобок. Символы – последовательность букв, цифр и знака $, могут быть любой длины, не могут начинаться с цифры. Символы могут служить переменными, в которых можно хранить любые значения, их можно использовать для создания индексированных переменных, которые по сути являются хэш-таблицами (hash tables). Отличие *Mathematica* от многих языков – символы могут не иметь никакого значения, т.е. могут

существовать сами по себе; символы можно рассматривать, как независимые объекты, а это – основа символьного программирования (например, как в диалектах LISP).

Выражение может быть представлено в виде математической формулы или ее части с помощью операторов, например a*(x+y+z) или x^y; оно может задавать и функцию f[x,y,...] или их комбинацию. Наряду с такой формой (математическая нотация), существует, так называемая, полная форма представления выражений, при которой основные арифметические операции задаются не операторами, а только соответствующими функциями системы.

Полная форма выражений.

Примеры полной формы основных математических выражений:

Выражение expr	Полная форма expr	Комментарий
x + y + z	Plus[x, y, z]	Сложение
x y z	Times[x, y, z]	Умножение
x^n	Power[x, n]	Возведение в степень
{a, b, c}	List[a, b, c]	Создание списка
a –> b	Rule[a, b]	Подстановка
a = b	Set[a, b]	Присваивание

В *Mathematica* перевод выражения expr в его полную форму обеспечивает функция FullForm[expr]. Ядро системы оперирует именно с представленными в полной форме выражениями. Пример:

```
FullForm[1 + x^2 - x / y + x * y - (y / z) ^3 + x / z]
```

```
Plus[1, Power[x, 2], Times[-1, x, Power[y, -1]], Times[x, y],
 Times[-1, Power[y, 3], Power[z, -3]], Times[x, Power[z, -1]]]
```

Для определения типа выражения служит функция Head[expr]. Применительно к числовым выражениям, она возвращает тип результата, при символьных выражениях функцию:

```
Head[1 + 2 + 3]
Head[123 / 456]
Head[123 / 456.]
Head[1 + 2 I]
Head[{1, 2, 3}]
Head[1 + x]
Head[(1 + x) ^2]
Head[x!]
```

```
Integer
```

```
Rational
```

```
Real
```

```
Complex

List

Plus

Power

Factorial
```

Следует особо отметить функцию Information, использование которой повышает скорость подготовки кода. Information предоставляет информацию о символе, функции, выводит формат записи:

```
Information[FullForm]
```

```
FullForm[expr] prints as the full form of expr, with no special syntax. >>

Attributes[FullForm] = {Protected}

Options[FullForm] = {NumberMarks :> True}
```

Части выражений, удаление элементов

Сложные выражения состоят из частей. Для выделения любой заданной части выражений используются функция Part или двойные квадратные скобки, 0-ая часть – тип, то что возвращает Head. Более подробно об этой функции будет изложение при изучении правил работы со списками, здесь отметим только основные возможности.

- Part[expr,m] выделяет m-ую часть выражения с начала, если m отрицательно, позиция отсчитывается с конца; эквивалентные записи: expr[[m]] (две подряд квадратные скобки), expr⟦…⟧ (двойные квадратные скобки);
- First[expr] – возвращает первый элемент в выражении;
- Last[expr] – возвращает последний элемент в выражении;
- Numerator[expr] – возвращает числитель выражения или указанной части;
- Denominator[expr] – возвращает знаменатель выражения или указанной части.

В примерах ниже следует обратить внимание на то, в каком порядке система запишет части выражения – это важно, чтобы знать номера элементов:

```
f3 := (a + b * x^2 + c * x^3) / (d + e * x^4) + g + h * x^2
f3 (* обратить внимание, как перепишет ...*)
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

```
f3
First[f3]
Head[f3]
Part[f3, 0]
Part[f3, 2]
f3[[3]]
Numerator[f3[[3]]]
Denominator[f3[[3]]]
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

g

Plus

Plus

$h\,x^2$

$$\frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

$a + b\,x^2 + c\,x^3$

$d + e\,x^4$

Если необходимо удалить части выражения, можно использовать следующие функции системы:

- Delete[expr,m] – удаляет в выражении expr элемент на позиции m (если m отрицательно, позиция отсчитывается с конца);
- Delete[expr,{i,j,...}] – удаляет части выражения на позициях {i,j,...};
- DeleteCases[expr,pattern] или DeleteCases[expr,pattern,levspec] – исключает все элементы выражения expr, которые отвечают шаблону (образцу) pattern; все или только на уровнях, указанных levspec (выражения могут быть многоуровневыми; уровень задается спецификацией levspec).

В примерах ниже следует обратить внимание, как надо перечислять удаляемые элементы, как записать шаблон (правило), чтобы удалить элемент x^2 (элементы обязательно перечислять внутренним списком, в случае удаления x^2 надо указать Infinity):

```
f4 := g + h * x^2 + (a + b * x^2 + c * x^3) / (d + e * x^4)
f4
Delete[f4, 2]
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

$$g + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

```
f4
Delete[f4, {1, 2}](* так не работает *)
Delete[f4, {{1}, {2}}]
DeleteCases[f4, x^2](* так не работает *)
DeleteCases[f4, x^2, Infinity]
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

... Delete: Part 2 of g does not exist.

$$\text{Delete}\left[g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}, \{1, 2\}\right]$$

$$\frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

$$g + h + \frac{a + b + c\,x^3}{d + e\,x^4}$$

Delete[expr,{i,j,...}] – удаляет части выражения на позициях {i,j,...}

```
f4
Delete[f4, {1, 2}(* так не работает *)]
Delete[f4, {2, 2}(* так работает, потому что во 2-ом ... *)]
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

... Delete: Part 2 of g does not exist.

$$\text{Delete}\left[g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}, \{1, 2\}\right]$$

$$g + h + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

DeleteCases[expr,pattern] или DeleteCases[expr,pattern,levspec] – исключает все элементы выражения expr, которые отвечают шаблону (образцу) pattern; все или только на уровнях, указанных levspec (выражения могут быть многоуровневыми; уровень задается спецификацией levspec):

```
f4
DeleteCases[f4, x^2](* так не работает (не указано ГДЕ)*)
DeleteCases[f4, x^2, Infinity]
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

$$g + h + \frac{a + b + c\,x^3}{d + e\,x^4}$$

Обратите внимание, что в f5 добавлено слагаемое с z^2:

```
f5 := g + h * x^2 + (a + b * z^2 + c * x^3) / (d + e * x^4)
DeleteCases[f5, x^2, Infinity]
```

$$g + h + \frac{a + c\,x^3 + b\,z^2}{d + e\,x^4}$$

```
f5 := g + h * x^2 + (a + b * z^2 + c * x^3) / (d + e * x^4)
(* ниже не конкретно x^2, а шаблон-конструкция x_^2 *)
DeleteCases[f5, x_^2, Infinity]
```

$$g + h + \frac{a + b + c\,x^3}{d + e\,x^4}$$

Уровни обеспечивают обычный способ ссылки на наборы частей в выражениях *Mathematica*. Чтобы указать уровень выражения, в системе используют спецификатор уровней выражения (levelspec). Вид спецификатора levelspec указывает уровни выражения согласно следующим правилам: n – с первого по n-й, {n} – только n-й, {m,n} – с m-го по n-й, Infinity – с первого по последний, Head→Trues – включая голову выражения, Head→False – не включая голову выражения. Можно представлять уровни в выражениях с точки зрения иерархической структуры. Уровень определенной части выражения является номером уровня вниз по иерархическому дереву до места расположения рассматриваемой части. Верхушка дерева принимается за уровень 0.

Для понимания уровня levspec конкретных элементов можно использовать:

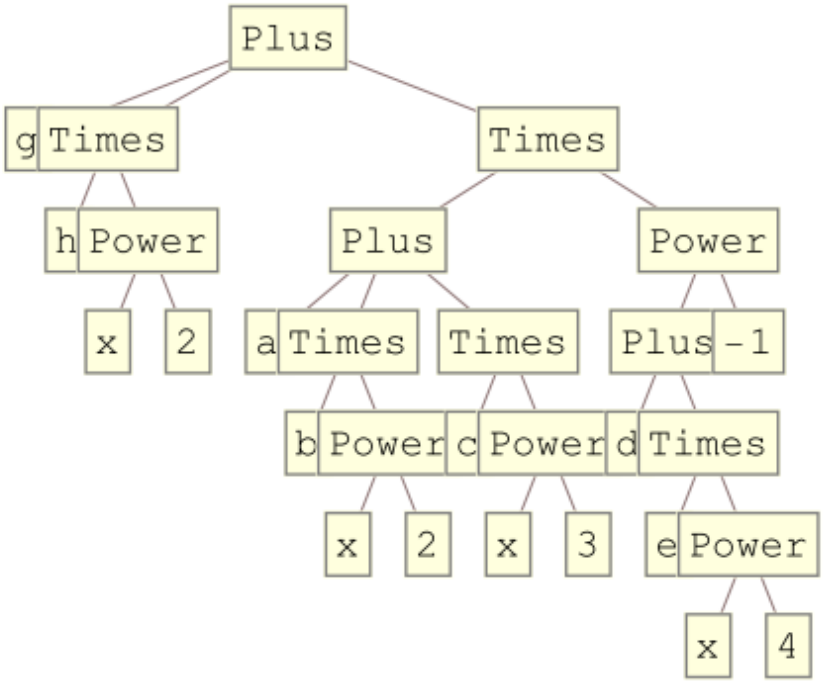
FullForm – полная форма (выражение в своей полной форме),

TreeForm – изображение в виде дерева (выражение в виде иерархической структуры).

Примеры:

```
FullForm[f4]
TreeForm[f4, AspectRatio → .8, ImageSize → 500]
```

```
Plus[g, Times[h, Power[x, 2]],
 Times[Plus[a, Times[b, Power[x, 2]], Times[c, Power[x, 3]]],
  Power[Plus[d, Times[e, Power[x, 4]]], -1]]]
```



Для определения позиций частей выражений используются: Position[expr,pattern], Position[expr,pattern,levspec].

Position[expr,pattern] – возвращает список позиций в expr, в которых размещаются объекты, сопоставимые с указанным шаблоном pattern. Примеры:

```
Position[f4, x^2]
Position[f4, x^2, 2]
Position[f4, x^2, 4]
Position[f4, x^4, 5]
```

```
{{2, 2}, {3, 1, 2, 2}}
```

```
{{2, 2}}
```

```
{{2, 2}, {3, 1, 2, 2}}
```

```
{{3, 2, 1, 2, 2}}
```

Position[expr,pattern,levspec] – выполняет поиск только объектов, находящихся на уровнях, указываемых levspec, начиная от первого и до уровня levspec:

```
f4
Position[f4, x^2, 2]
Position[f4, x^2, 4]
```

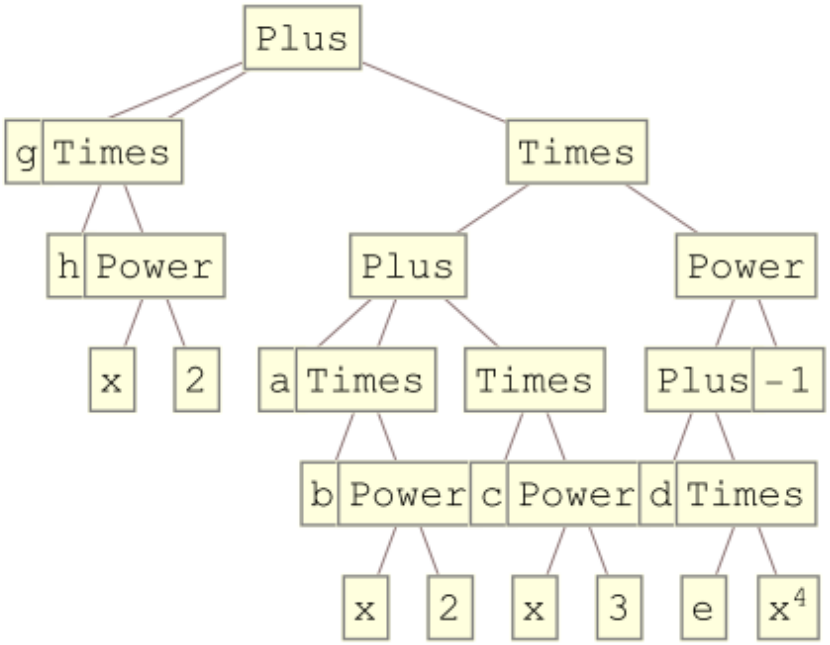
$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

```
{{2, 2}}
```

```
{{2, 2}, {3, 1, 2, 2}}
```

```
f4
TreeForm[f4, 5, AspectRatio → .75, ImageSize → 500]
(* первые 5 уровней выражения *)
Position[f4, x^4, 5]
(* ниже демонстрация, что на 4 уровне такого нет *)
Position[f4, x^4, 4]
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$




```
{{3, 2, 1, 2, 2}}
```

```
{}
```

Position[f4,x^_,5]. *Обратите внимание*, что здесь x в любой степени на всех уровнях до 5-го:

```
f4
Position[f4, x^_, 5]
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

```
{{2, 2}, {3, 1, 2, 2}, {3, 1, 3, 2}, {3, 2, 1, 2, 2}}
```

Пример Position[f4,x^_,{5}]. *Обратите внимание*, что здесь x в любой

степени на уровне 5

```
f4
Position[f4, x^_, {5}]
```

$$g + h\,x^2 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

```
{{3, 2, 1, 2, 2}}
```

Общее о шаблонах. Шаблон/образец/ (pattern) в *Mathematica* – выражение, которое описывает некоторое множество выражений. В выражении, являющемся шаблоном, обязательно содержится символ подчеркивания _, один или несколько. Произвольное выражение в *Mathematica* описывается одним символом подчеркивания _. Полная форма этого символа Blank[ ].

Одно или несколько выражений описывают, используя два символа подчеркивания __, полная форма – BlankSequence[ ].

Самое широкое множество выражений описывается с помощью трех символов подчеркивания ___. Так указывается или пустое множество, или одно выражение, или несколько выражений. Полная форма этого выражения – BlankNullSequence[ ].

С помощью шаблонов можно описывать множества выражений, обладающих определенными свойствами. Эти выражения, например, имеют одинаковую голову, или одинаково реагируют на тест-функцию, встроенную или созданную пользователем.

Примеры шаблонов даны ниже (включены наиболее часто употребляемые Integer, Real, Rational, Complex, Symbol, List).

Построение шаблонов является непростой задачей. На помощь приходит встроенная функция MathQ[expression, pattern], которая проверяет данное выражение expression на соответствие указанному образцу pattern.

Они используются в *Mathematica* для выделения классов выражений (отбор по заданным условиям), преобразования выражений из одной формы в другую, при работе со списками. Есть типовые, а также количество шаблонов можно расширить за счёт создания собственных пользовательских правил.

Самый широкий класс выражений очерчивает простейший шаблон, состоящий только из одного символа подчёркивания _. Этому классу соответствуют абсолютно все выражения *Mathematica*. Например, чтобы задать шаблон для действительных чисел, следует записать _Real, а чтобы выделить в отдельный класс все символьные выражения – задать _Symbol. Можно объединить в один класс выражения, обладающие определённым

заголовком name – для этого задаётся шаблон вида _name.

Основные шаблоны. *Mathematica* содержит набор встроенных функций для тестирования свойств выражений. Буква Q в конце названия таких функций служит своеобразным условным обозначением, указывающим на то, что они "задают вопрос".

IntegerQ[expr] – целое число, EvenQ[expr] – четное число, OddQ[expr] – нечетное число, PrimeQ[expr] – простое число, NumberQ[expr] – число любого рода, NumericQ[expr] – численная величина, PolynomialQ[expr,{x_1,x_2,...] – многочлен в переменных x_1,x_2, ..., , VectorQ[expr] – список представляющий вектор, ArrayQ[expr,d] – полный массив с глубиной равной d, MatrixQ[expr] – вложенный список представляющий матрицу, .

Для того, чтобы определить относится ли выражение expr к классу данных, задаваемому шаблоном pattern, следует воспользоваться функцией

- MatchQ[expr,pattern] – возвращает значение True, если выражение expr соответствует шаблону pattern, и False, если не соответствует.

Для определения позиций частей выражений используются:

- Position[expr,pattern] – возвращает список позиций в expr, в которых размещаются объекты, сопоставимые с указанным шаблоном pattern;
- Position[expr,pattern,levspec] – выполняет поиск только объектов, находящихся на уровнях, указываемых levspec.

Примеры шаблонов (включены наиболее часто употребляемые Integer, Real, Rational, Complex, Symbol, List):

```
MatchQ[12345, _Integer]
MatchQ[12345., _Integer]
MatchQ[12345 / 13, _Integer]
```

```
True

False

False
```

```
MatchQ[12345., _Real]
MatchQ[Sqrt[2], _Rational]
```

```
True

False
```

```
MatchQ[Sqrt[2], _Complex]
MatchQ[123 + 3.45 I, _Complex]
MatchQ[123 + 3.45 I, _Symbol]
```

```
False
```

```
True

False
```

```
MatchQ[a123, _Symbol]
MatchQ[12345., _List]
```

```
True

False
```

```
MatchQ[{12345., 12345 / 13}, _List]
```

```
True
```

Манипуляции с выражениями

Кроме удаления возможны другие изменения выражений. Наиболее часто используются следующие функции:

- Append[expr,elem] – возвращает expr с дополнением elem;
- AppendTo[s,elem] – добавляет elem к значению s и переустанавливает s в новое значение;
- Apply[f,expr,levelspec] – возвращает expr, замещая заголовки в тех частях expr, которые указаны спецификацией уровня levelspec;
- Cancel[expr] – возвращает expr с сокращением общих множителей числителя и знаменателя;
- Cases[expr,pattern,levelspec] – возвращает список всех частей выражения expr на уровнях, указанных спецификацией levelspec, и которые соответствуют шаблону pattern;
- Chop[expr] – в комплексном выражении expr задает нулевой малую мнимую часть;
- Chop[expr,err] – присваивает значение 0 приближенным вещественным числам в выражении expr, абсолютные величины которых меньше err;
- Replace[expr,rules] – возвращает expr с подстановкой заданной правилом или списком правил rules;
- ReplaceAll – используется в виде: expr/.rules – возвращает expr с подстановками, заданными правилом или списком правил;
- ReplaceHeldPart[expr,Hold[new], n] – возвращает выражение, в котором n-ная часть expr заменена на new;
- ReplaceHeldPart[expr,Hold[new],{i,j,...}] – заменяет часть на позиции {i,j,...};
- ReplaceHeldPart[expr,Hold[new],{{i1,j1,...}, {i2,j2,...},...}] – замещает части в нескольких позициях на new;
- ReplacePart[expr,new,n] – возвращает выражение, в котором n-ная часть expr замещена на new;

- ReplacePart[expr,new,{i,j,...}] – замещает часть на позиции {i,j,...};
- ReplacePart[expr,new,{{i1,j1,...},{i2,j2,...},...}] – замещает части в нескольких позициях на new;
- ReplaceRepeated – применяется в виде: expr//.rules – неоднократно выполняет замещения до тех пор, пока expr не перестанет изменяться.

Примеры:

```
f4 := g + h * x^2 + (a + b * x^2 + c * x^3) / (d + e * x^4)
Append[f4, u * x^5]
ReplacePart[f4, x^(3 / 2), 2]
```

$$g + h\,x^2 + u\,x^5 + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

$$g + x^{3/2} + \frac{a + b\,x^2 + c\,x^3}{d + e\,x^4}$$

```
Cancel[123 456 / 12 345 + h * x^2]
f5 = Expand[(a + b * x^2)^3]
Cancel[f5 / (a + b * x^2)]
```

$$\frac{41\,152}{4115} + h\,x^2$$

$$a^3 + 3\,a^2\,b\,x^2 + 3\,a\,b^2\,x^4 + b^3\,x^6$$

$$a^2 + 2\,a\,b\,x^2 + b^2\,x^4$$

Обычная функция **ReplaceAll** (короткая форма записи /.) – заменить всё, предназначена для того, чтобы применить предложенное правило замены (один раз). **ReplaceRepeated** (короткая форма записи //.) – заменить с повторением, использует указанное правило до тех пор, пока выражение не перестанет меняться, т.е. не будет более возможности правило применить.

Примеры:

```
x^2 + y^6 /. {x → 2 + a, a → 3}
x^2 + y^6 //. {x → 2 + a, a → 3}
```

$$(2 + a)^2 + y^6$$

$$25 + y^6$$

Condition (сокращенная форма записи /;) – условие. **lhs:>rhs/;test** представляет правило, которое применяется только в том случае, если оценка теста дает True.

Пример замены всех элементов, удовлетворяющих условию отрицательности:

```
{6, -17, 3, 2, -13, -2} /. x_ /; x < 0 → s
```

```
{6, s, 3, 2, s, s}
```

p.. or **Repeated[p]** – является объектом шаблона, представляющим последовательность одного или нескольких выражений, каждое из которых соответствует p.

Пример ниже – замена в списке всех элементов из нескольких a на x:

```
{{a, b, c}, {a, a}, {a, b}, {a, a, a}, {a, c}, {a}} /. {a ..} → x
```

```
{{a, b, c}, x, {a, b}, x, {a, c}, x}
```

Функции. Формы записи

В системе *Mathematica* функции характеризуются именем, определяющим функциональную зависимость выражением, задаваемым в квадратных скобках списком параметров (аргументов), который может быть пустым. Понятие функции ассоциируется с возвратом некоторого объекта в ответ на обращение к ней по ее имени и при указании аргументов. В ответ на обращение функция возвращает значение выражения – численное или символьное, также может возвратить и объект, например графический, звуковой, мультимедиа. Возврат функциями объектов-результатов позволяет применять их наряду с операторами для составления математических выражений.

В основном входной язык общения с системой *Mathematica* ориентирован на принципы функционального программирования с применением полных форм представления выражений, но также могут реализовываться сокращённые формы и все парадигмы программирования о чём будут пояснения и примеры отдельно.

Язык системы *Mathematica* поддерживает все формы записи выражений: стандартную; префиксную, в которой оператор расположен перед операндами; постфиксную, в которой оператор расположен после операндов; инфиксную, в которой оператор расположен между операндами. Постфиксная и префиксная формы образуют т.н. польскую форму записи (польская нотация). Польская форма удобна, прежде всего, тем, что в ней отсутствуют скобки.

Приведём названные формы записи выражений:

- f[x,y] – стандартная форма для f[x,y],
- f@x – префиксная форма для f[x],
- x//f – постфиксная форма для f[x],
- x~u~y – инфиксная форма для u[x,y].

Замечание. Дополнительные сведения о применении укороченной формы функций описаны ниже в примерах работы с функциями Apply, Map, MapAll.

Примеры вывода:

```
Postfix[fEx1[x]]
Prefix[fEx1[x]]
Infix[fEx2[x, y, z], "~"]
```

```
x // fEx1
```

```
fEx1@x
```

```
x ~ y ~ z
```

Примеры применения упомянутых форм при записи функций:

```
f1[x_] := 1 + 2 * x^2 + 3 * x^3
f1[a]
f1@a
a // f1
```

$$1 + 2\,a^2 + 3\,a^3$$

$$1 + 2\,a^2 + 3\,a^3$$

$$1 + 2\,a^2 + 3\,a^3$$

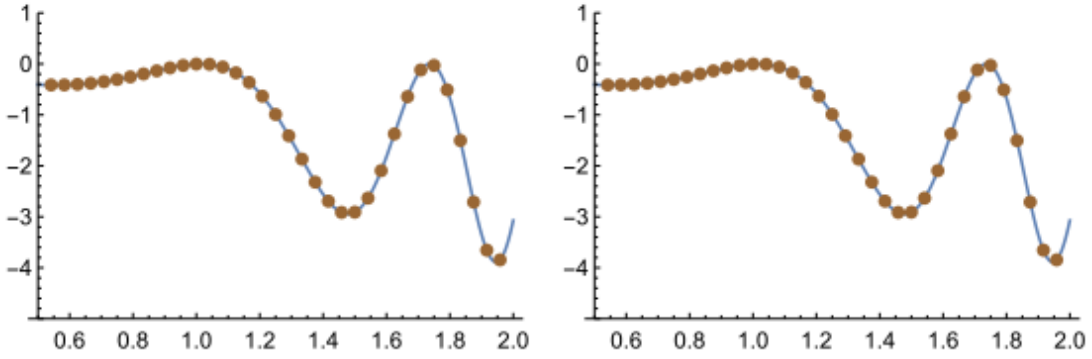
```
f2[x_, y_] := x^3 - y^2
f2[a, b]
```

$$a^3 - b^2$$

Примеры применения упомянутых форм при записи директив функций графики, варианты:

```
f1[x] := Sin[1.5 * x^3];
f2[x] := Cos[9 * x^3];

grf3a = Plot[x * f1[x] - x, {x, 0.5, 2.}, PlotRange -> {1, -5},
    BaseStyle -> 12, AxesOrigin -> {0.5, -5},
    Mesh -> 35,
    MeshStyle -> Directive[Brown, PointSize[Large]],
    ImageSize -> 275];
grf3b = Plot[x * f1[x] - x, {x, 0.5, 2.}, PlotRange -> {1, -5},
    BaseStyle -> 12, AxesOrigin -> {0.5, -5},
    Mesh → 35,
    MeshStyle -> Directive@{Brown, PointSize@Large},
    ImageSize -> 275];
(* Row - ряд, Spacer - разделительный пропуск *)
Row[{grf3a, grf3b}, Spacer[10]]
```



Встраиваемые функции.

Возможность определять и использовать собственные функции – это то, что существенно расширяет и упрощает программирование в среде *Mathematica*. Однако, код будет перегружен, если всем функциям задавать имена. Система позволяет объявлять функции встраиваемыми (их называют чистыми функциями), и обойти это неудобство. Простейшим способом определить чистую функцию является использование Function. Варианты:

- Function[body] – создает чистую функцию с телом body;
- Function[{x},body] – создает чистую функцию параметра x с телом body;
- Function[{x1,x2,...},body] – создает чистую функцию ряда параметров x1, x2,... с телом body.

Для вычисления чистой функции после неё надо ввести в квадратных скобках список параметров. Например, для взятой в качестве примера функции ее можно задать и использовать следующим образом:

```
Function[{x}, 1 + 2 * x^2 + 3 * x^3]
%[a]
%%[a + b]
```

$$\text{Function}\left[\{x\}, 1 + 2\,x^2 + 3\,x^3\right]$$

$$1 + 2\,a^2 + 3\,a^3$$

$$1 + 2\,(a + b)^2 + 3\,(a + b)^3$$

Примеры использования чистой функции с несколькими аргументами:

```
Function[{x, y, z}, x + y - z]
%[a, b, c]
```

```
Function[{x, y, z}, x + y - z]
```

```
a + b - c
```

```
Function[{x, y, z}, x - y + z][d, e, f]
```

```
d - e + f
```

Чистую функцию просто превратить в функцию пользователя, что

показывает следующий пример:

```
fRab = Function[{x}, 1 + 2 * x^2 + 3 * x^3]
fRab[a]
```

```
Function[{x}, 1 + 2 x^2 + 3 x^3]

1 + 2 a^2 + 3 a^3
```

Анонимные функции

Предельно компактную форму задания имеют так называемые анонимные функции. Они не имеют ни названия, ни обычного определения и задаются только выражениями специального вида. В такой форме в выражении вместо переменных используют обозначения # (для одной переменной) или #1, #2, ... (для нескольких переменных). Завершается тело функции символом &. Если надо вычислить функцию, то после ее записи в квадратных скобках указывается список фактических параметров. Преимущество анонимной формы записи функции заключается в том, что для функции не требуется отдельного определения или имени.

Анонимная функция в программировании – особый вид функций, которые объявляются в месте использования и не получают уникального идентификатора для доступа к ним. Поддерживаются во многих языках программирования. Обычно при создании анонимные функции либо вызываются напрямую, либо ссылка на функцию присваивается переменной, с помощью которой затем можно косвенно вызывать данную функцию. Анонимные функции имеют предельно компактную форму задания. Они не имеют ни названия, ни обычного определения и задаются только выражениями специального вида. В такой форме в выражении вместо переменных используют обозначения # (для одной переменной) или #1, #2, ... (для нескольких переменных). Завершается тело функции символом &. Если надо вычислить функцию, то после ее записи в квадратных скобках указывается список фактических параметров. Преимущество анонимной формы записи функции заключается в том, что для функции не требуется отдельного определения или имени.

Простыми словами: # – отдельно не используется, аргумент функции.
Простыми словами: & – отдельно не используется, отделяет функцию от остального выражения

Пример: #^2& – аналогично Function[x, x^2], что создает функцию. Другими словами – #^2&[z] вернет z^2.

Замечание. Анонимные (безымянные) функции применяются во многих языках программирования, например, Common Lisp, Python, PHP, C#, F#, Visual Basic .NET, C++, Scala, Java (их также называют

лямбда-выражениями).

Для рассмотренного выше примера Function[{x,y,z},x-y+z][d,e,f] анонимная функция и вычисление выглядит так:

```
#1 - #2 + #3 &[d, e, f]
```

```
d - e + f
```

С помощью анонимных функций нетрудно создавать обычные функции пользователя:

```
f3 = (1 + 2 * #^2 + 3 * #^3) &;
f3[a]
```

$$1 + 2\,a^2 + 3\,a^3$$

Резюмируя, напомним, что один и тот же результат вычислений дадут рассмотренные для выражения 1+2*x^2+3*x^3 варианты записи:

```
f1[x_] := 1 + 2 * x^2 + 3 * x^3
f1[a]
f1@a
a // f1
f3 = (1 + 2 * #^2 + 3 * #^3) &;
f3[a]
(1 + 2 * #^2 + 3 * #^3) &[a]
```

Также для примера приведём первоначально записанный и дополнительный (сокращённый, с использованием @, #, &) варианты записи кода для получения и вывода списка “однобуквенных” функций системы:

```
Select[Names["System`*"],
 Function[{smb}, StringLength[ToString[FullForm[smb]]] == 3]]
Select[Names["System`*"],
 StringLength[ToString[FullForm@#]] == 3 &]
```

```
{C, D, E, I, K, N, O}
```

```
{C, D, E, I, K, N, O}
```

Замечание. Применение анонимных функций открывает возможности компактного задания, но эта форма едва ли интересна для начинающих пользователей – многие, конечно же, предпочтут пусть более длинное, но простое и понятное задание функций. Есть примеры, когда код нельзя записать без анонимных функций. Также надо принимать во внимание, что традиционное использование функций предполагает, что имя и обращение к функции говорят “что” она делает, но при желании скрывается “как” – у анонимных всё наоборот: имени нет, явно записано как выполняется.

Обращение к функции, получение результата.

Выше отмечены возможные варианты записи и выполнения функций f[expr] (префиксная форма – f@expr), Apply[f,expr] (f@@expr). Дополнительно приведём эквивалентные записи: f@@{a,b,c} – f[a,b,c]; f/@{a,b,c} – {f[a],f[b],f[c]}; f//@{a,b,x} – f[{f[a],f[b],f[x]}].

Отдельно отметим назначение и примеры применения функций Nest, Map, MapAll:

- Nest[f,expr,m] – применяет функцию и возвращает выражение, полученное m-кратным приложением f к expr;
- NestList[f,expr,m] – возвращает список приложений функции f к выражению expr m раз;
- Map[f,expr] – прикладывает f к каждому элементу на первом уровне в expr (укороченная форма – f/@expr);
- Map[f,expr,levelspec] – применяет f к частям expr уровня levelspec;
- MapAll[f,expr] – прикладывает f ко всем частям выражения expr (укороченная форма – f//@expr).

Примеры применения :

```
Nest[f6, x, 3]
NestList[f6, x, 3]
```

```
f6[f6[f6[x]]]
```

```
{x, f6[x], f6[f6[x]], f6[f6[f6[x]]]}
```

```
Nest[Sqrt, 10000.0, 3]
NestList[Sqrt, 10000.0, 3]
```

```
3.16228
```

```
{10000., 100., 10., 3.16228}
```

```
Nest[#^a &, x, 3]
NestList[#^a &, x, 3]
```

$$\left(\left(x^a\right)^a\right)^a$$

$$\left\{x,\ x^a,\ \left(x^a\right)^a,\ \left(\left(x^a\right)^a\right)^a\right\}$$

```
Nest[a^# &, x, 2]
NestList[a^# &, x, 2]
```

$$a^{a^x}$$

$$\left\{x,\ a^x,\ a^{a^x}\right\}$$

```
Nest[(1 + #)^2 &, a, 3]
NestList[(1 + #)^2 &, x, 3]
```

$$\left(1 + \left(1 + (1 + a)^2\right)^2\right)^2$$

$$\left\{x, (1 + x)^2, \left(1 + (1 + x)^2\right)^2, \left(1 + \left(1 + (1 + x)^2\right)^2\right)^2\right\}$$

```
Nest[(1 + 2 * #)^2 &, a, 3]
NestList[(1 + 2 * #)^2 &, a, 3]
```

$$\left(1 + 2\left(1 + 2\left(1 + 2\,a\right)^2\right)^2\right)^2$$

$$\left\{a, (1 + 2\,a)^2, \left(1 + 2\,(1 + 2\,a)^2\right)^2, \left(1 + 2\left(1 + 2\,(1 + 2\,a)^2\right)^2\right)^2\right\}$$

```
Map[f7, {a, b, c}]
MapAll[f7, {a, b, c}]
```

```
{f7[a], f7[b], f7[c]}
```

```
f7[{f7[a], f7[b], f7[c]}]
```

```
Map[f8, a + b * x]
MapAll[f8, a + b * x]
```

```
f8[a] + f8[b x]
```

```
f8[f8[a] + f8[f8[b] f8[x]]]
```

Примеры использования анонимных функций в 2D, 3D визуализации

В изложении материалов текущей темы, как и предыдущей, используется выражение $zXY(x,y)$, определяемое аналитически:

$$zXY(x, y) = e^{-(x-9)^2-(y-2)^2} - 2/3\, e^{-(x-7)^2-(3y-3)^2}$$

Здесь вводим и визуализируем функцию $zXY(x,y)$. Внимание! Не ставим целью качественно, максимально просто, т.к. цель – показать применение в опциях # и &.

```
xMin := 5; xMax := 11; yMin := 0; yMax := 4;
aspRat = (yMax - yMin) / (xMax - xMin);
zXY[x_, y_] :=
  E^(-(x - 9)^2 - (y - 2)^2) - (2 / 3) * E^(-(x - 7)^2 - (3 * y - 3)^2);
zXYmin =
 NMinimize[{zXY[x, y], xMin < x < xMax, yMin < y < yMax}, {x, y}]
zXYmax =
 NMaximize[{zXY[x, y], xMin < x < xMax, yMin < y < yMax}, {x, y}]
```

```
{-0.66019, {x → 6.98112, y → 0.99896}}
```

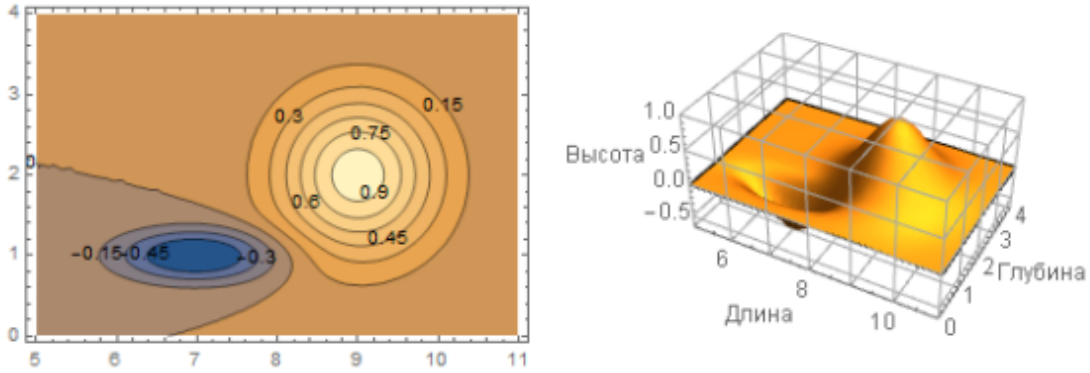
```
{0.999998, {x → 9., y → 2.00001}}
```

Визуализация в вариантах 2D и 3D с минимумом настроек, ContourPlot – контурный график, Plot3D – график функции 2-х переменных:

```
grf2d0 = ContourPlot[zXY[x, y], {x, xMin, xMax},
    {y, yMin, yMax}, PlotRange → Full, AspectRatio → aspRat,
    Contours → 10, ContourLabels → All, ImageSize → 275];

zWINmin = -2 / 3; zWINmax = 1; zScale = 3;
grf3d0 = Plot3D[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    PlotRange → {{xMin, xMax}, {yMin, yMax}, {zWINmin, zWINmax}},
    BaseStyle → {Gray, 12}, Mesh → None,
    AxesLabel → {"Длина", "Глубина", "Высота"}, FaceGrids → All,
    BoxRatios → {xMax - xMin, yMax - yMin, zScale}, ImageSize → 275];

Row[{grf2d0, grf3d0}, Spacer[10]]
```

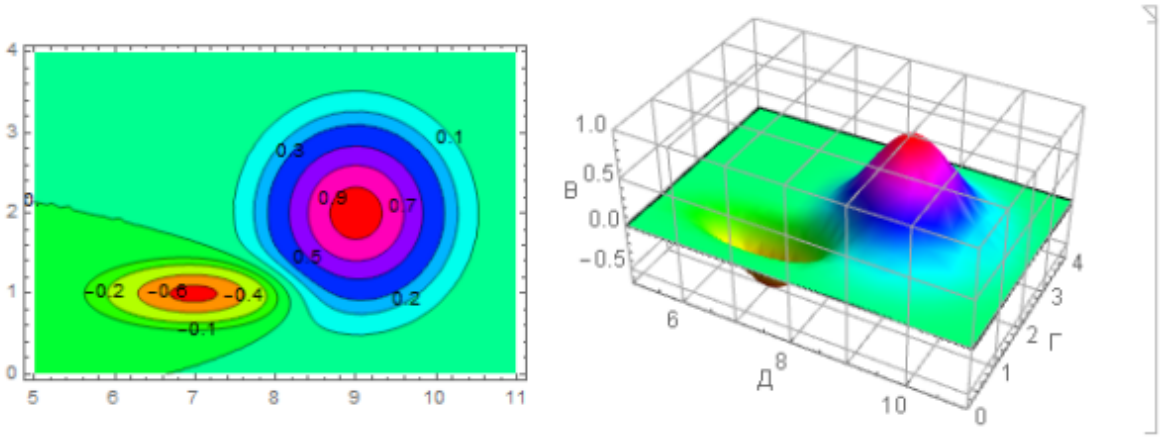


Ниже иллюстрации даются с опцией Contours→zIZL, в которой задаются значения уровней выводимых контурных линий:

```
zIZL = {-0.6, -0.4, -0.2, -0.1, 0.0, 0.1, 0.2, 0.3, 0.5, 0.7, 0.9};
grf2d1 = ContourPlot[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    PlotRange → Full, AspectRatio → aspRat, Contours → zIZL,
    ContourLabels → All, ColorFunction → Hue, ImageSize → 275];

grf3d1 = Plot3D[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    PlotRange → {{xMin, xMax}, {yMin, yMax}, {zWINmin, zWINmax}},
    BaseStyle → {Gray, 12}, Mesh → None, AxesLabel → {"Д", "Г", "В"},
    FaceGrids → All, BoxRatios → {xMax - xMin, yMax - yMin, zScale},
    ColorFunction → Hue, ImageSize → 275];

Row[{grf2d1, grf3d1}, Spacer[10]]
```



Использование анонимных функций в 2D визуализации. График grfMesh1 дает представление, как вывести только требуемые линии сетки в области визуализации – Mesh→{6,3}. Конкретно, в исполняемой секции записаны число линий и указано по каким переменным сколько – {6,3}, {#1&,#2&}, также задается стиль линий сетки – MeshStyle→ Directive[Dashed,Gray]:

```
grfMesh1 = ContourPlot[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    AspectRatio → aspRat, PlotRange → Full,
    Contours → zIZL, ContourLabels → All,
    Mesh → {6, 3}, MeshStyle → Directive[Dashed, Gray],
    MeshFunctions → {#1 &, #2 &},
    ColorFunction → Hue, ImageSize → 275];
Row[{grf2d1, grfMesh1}, Spacer[10]]
```

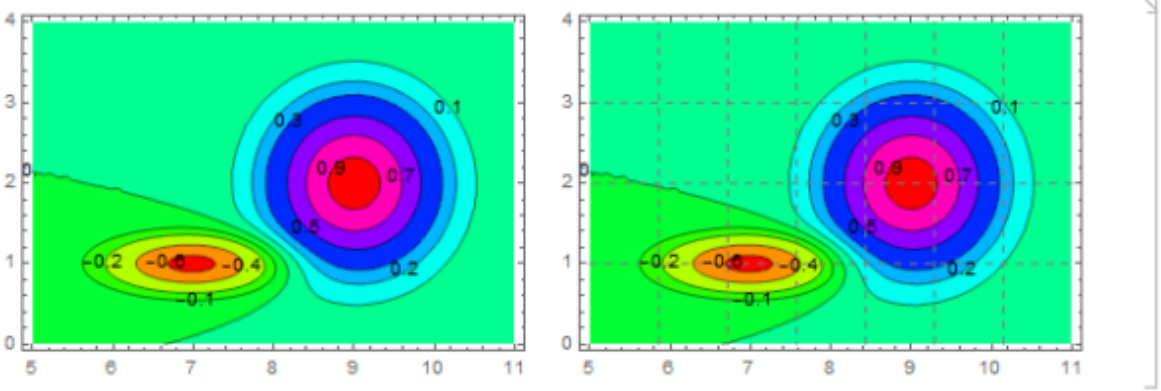
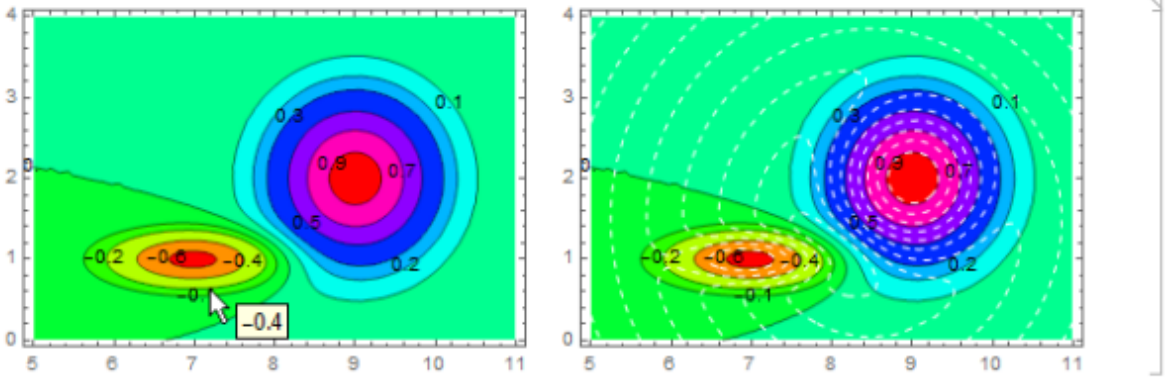


График grfMesh2 (пример довольно экзотический, но подобные встречаются в практике анализа и интерпретации цифровых распределений) дает представление, как вывести задуманные линии в области визуализации – MeshFunctions→{**Norm[{(#1-8.5),(#2-1.5),#3*5}]&**}. Конкретно, здесь выводятся 9 уровней на цифровом поле, когда принята норма, где отсчет расстояния идет от точки (8.5,1.5) и вес z-значения функции (высоты поверхности) усилен в 5 раз:

```
grfMesh2 = ContourPlot[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    AspectRatio → aspRat, PlotRange → Full,
    Contours → zIZL, ContourLabels → All,
    MeshFunctions → {Norm[{(#1 - 8.5), (#2 - 1.5), #3 * 5}] &},
    Mesh → 9, MeshStyle → Directive[Dashed, White],
    ColorFunction → Hue,
    ImageSize → 275];

Row[{grf2d1, grfMesh2}, Spacer[10]]
```



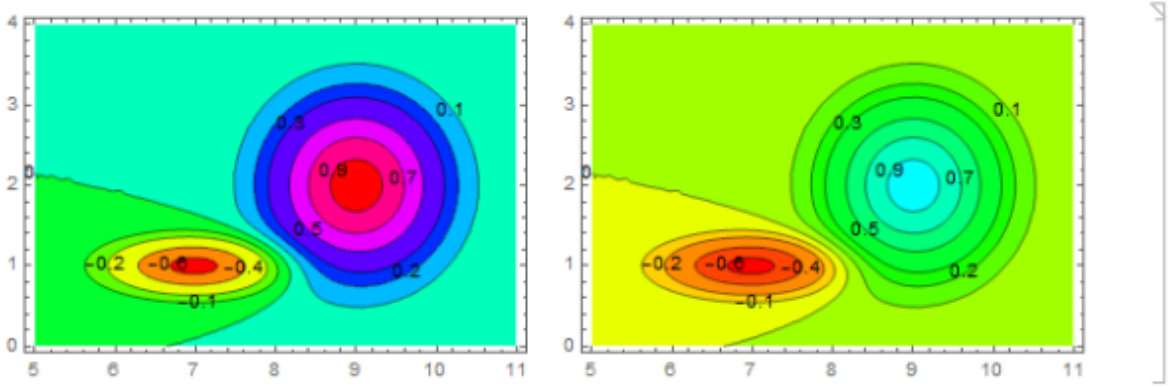
Графики grfShad1 и grfShad2 дают представление, как задавать цвета заливки участков между изолиниями (затенение областей между контурными линиями задает ContourShading). В примерах при задании цветов раскраски использованы весь (Hue/@Range[0,1.0,1/nIZL]) и половина цветового круга Hue (Hue/@ Range[0.,0.5,0.5/nIZL):

```
nIZL = Length[zIZL];

grfShad1 = ContourPlot[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    AspectRatio → aspRat, PlotRange → Full,
    Contours → zIZL, ContourLabels → All,
    ContourShading → Hue /@ Range[0, 1.0, 1 / nIZL],
    ImageSize → 275];

grfShad2 = ContourPlot[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    AspectRatio → aspRat, PlotRange → Full,
    Contours → zIZL, ContourLabels → All,
    ContourShading → Hue /@ Range[0., 0.5, 0.5 / nIZL],
    ImageSize → 275];

Row[{grfShad1, grfShad2}, Spacer[10]]
```



Использование анонимных функций в 3D визуализации. Графики grf3dMesh1 и grf3dMesh2 иллюстрируют возможности визуализации с разными вариантами использования опции Mesh.

Вывод 10 линий уровней функции z (поверхности) обеспечивают установки MeshFunctions→{#3&}, Mesh→10.

График grf3dMesh2 иллюстрирует возможности дополнения вида формы поверхности координатными линиями. Установки MeshFunctions→{#1&,#2&}, Mesh→{6,4} обеспечивают вывод 6×4 линий в направлениях 0X, 0Y; опция MeshStyle→Directive[Dashed, Black] используется для задания стиля этих линий: :

```
grf3dMesh1 = Plot3D[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    PlotRange → {{xMin, xMax}, {yMin, yMax}, {zWINmin, zWINmax}},
    BaseStyle → {Gray, 12},
    AxesLabel → {"Д", "Г", "В"}, FaceGrids → All,
    BoxRatios → {xMax - xMin, yMax - yMin, zScale},
    ColorFunction → Hue,
    MeshFunctions → {#3 &}, Mesh → 10,
    ImageSize → 275];

grf3dMesh2 = Plot3D[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    PlotRange → {{xMin, xMax}, {yMin, yMax}, {zWINmin, zWINmax}},
    BaseStyle → {Gray, 12},
    AxesLabel → {"Д", "Г", "В"}, FaceGrids → All,
    BoxRatios → {xMax - xMin, yMax - yMin, zScale},
    ColorFunction → Hue,
    MeshFunctions → {#1 &, #2 &}, Mesh → {6, 4},
    MeshStyle → Directive[Dashed, Black],
    ImageSize → 275];

Row[{grf3dMesh1, grf3dMesh2}, Spacer[10]]
```

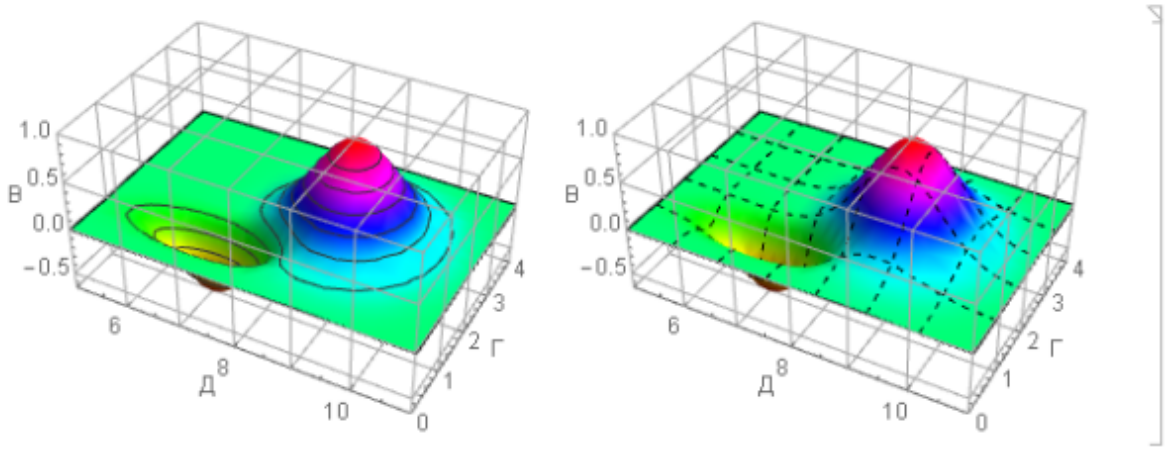


График grf3dMesh3 иллюстрирует возможности лоскуткового представления вида формы поверхности. Установки Mesh→{13,7}, MeshFunctions→{#1&,#2&}обеспечивают дискретность 13×7; опция MeshShading→{{Cyan,Yellow},{Yellow,Cyan}} используется для задания стиля лоскутков, опция MeshStyle→Directive[Dashed,Red] используется для задания стиля границ лоскутков:

```
grf3dMesh3 = Plot3D[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    PlotRange → {{xMin, xMax}, {yMin, yMax}, {zWINmin, zWINmax}},
    BaseStyle → {Gray, 12}, AxesLabel → {"Д", "Г", "В"},
    FaceGrids → All, BoxRatios → {xMax - xMin, yMax - yMin, zScale},
    ColorFunction → Hue,
    Mesh → {13, 7}, MeshFunctions → {#1 &, #2 &},
    MeshShading → {{Cyan, Yellow}, {Yellow, Cyan}},
    MeshStyle → Directive[Dashed, Red],
    ImageSize → 275];

Row[{grf3d1, grf3dMesh3}, Spacer[10]]
```

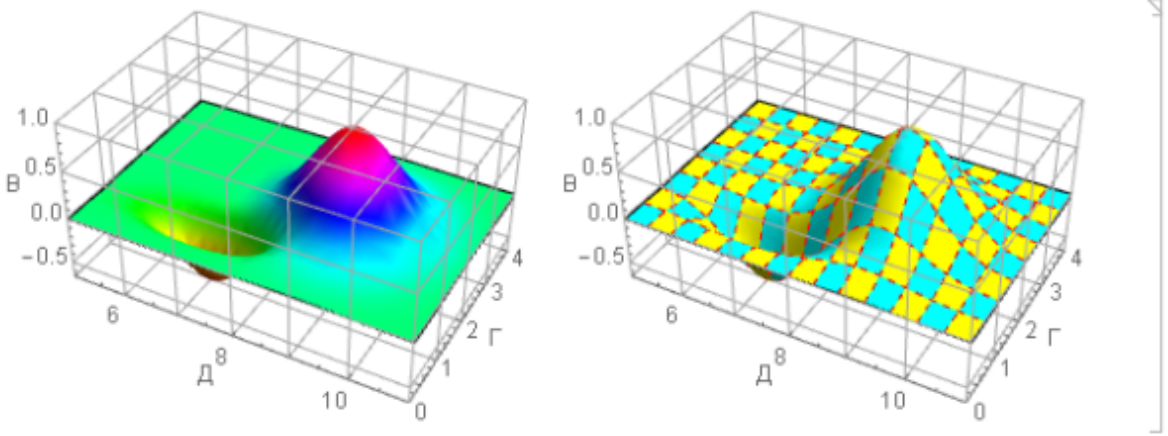


График grf3dMesh4 иллюстрирует возможности дополнения вида формы поверхности линиями, точки которых равноудалены от выбранного пункта (8,2,0). Установки MeshFunctions→{Sqrt[(#1-8)^2+(#2-2)^2+#3^2]&},

Mesh→7 обеспечивают вывод 7 линий; опция MeshStyle→ Directive[Dotted,Purple] используется для задания стиля этих линий:

```
grf3dMesh4 = Plot3D[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    PlotRange → {{xMin, xMax}, {yMin, yMax}, {zWINmin, zWINmax}},
    BaseStyle → {Gray, 12}, AxesLabel → {"Д", "Г", "В"},
    FaceGrids → All, BoxRatios → {xMax - xMin, yMax - yMin, zScale},
    ColorFunction → Hue,
    MeshFunctions -> {Sqrt[(#1 - 8)^2 + (#2 - 2)^2 + #3^2] &},
    Mesh → 7, MeshStyle → Directive[Dotted, Purple],
    ImageSize → 275];

Row[{grf3d1, grf3dMesh4}, Spacer[10]]
```

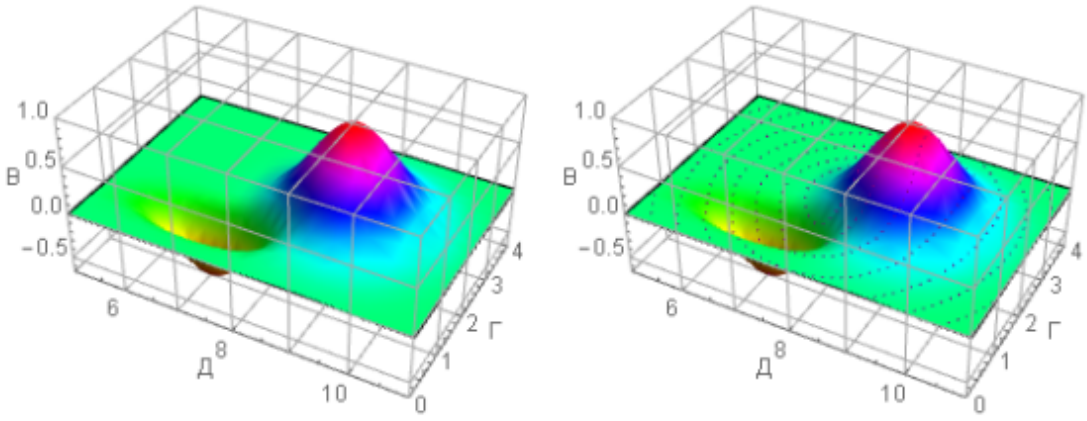
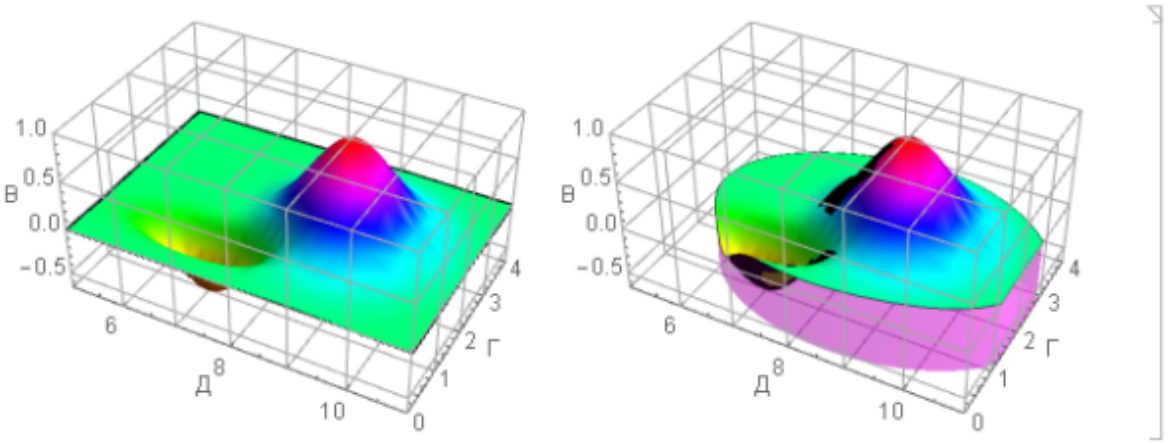


График grf3dRegion иллюстрирует возможности вывода на участке, ограниченном по периметру, с задаваемым уравнением границы – ((#1-8.5)^2+2.5*(#2-2)^2<8&). Установки RegionFunction→ (#1-8.5)^2+2.5*(#2-2)^2<8&) обеспечивают вывод на участке, а не во всей области определения; опции Filling→Bottom, FillingStyle→ Directive[Opacity[0.3],Magenta] используются для задания стиля области отсечения:

```
grf3dRegion = Plot3D[zXY[x, y], {x, xMin, xMax}, {y, yMin, yMax},
    PlotRange → {{xMin, xMax}, {yMin, yMax}, {zWINmin, zWINmax}},
    BaseStyle → {Gray, 12}, AxesLabel → {"Д", "Г", "В"},
    FaceGrids → All, BoxRatios → {xMax - xMin, yMax - yMin, zScale},
    ColorFunction → Hue, Mesh → None,
    RegionFunction -> ((#1 - 8.5)^2 + 2.5 * (#2 - 2)^2 < 8 &),
    Filling → Bottom,
    FillingStyle → Directive[Opacity[0.3], Magenta],
    ImageSize → 275];

Row[{grf3d1, grf3dRegion}, Spacer[10]]
```



Тестовые задания для самоконтроля

Упражнение ВФ-1.

***Mathematica*. Синтаксис языка. Использование анонимных функций.**

В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в 3-й строке. Воспроизвести.

```
a=11;b=9;
???[a]
2357947691
```

Отправить, что должно быть вместо ???.

Подсказки: Символов в ответе - 4, есть b. Пробелов в ответе быть не должно

Упражнение ВФ-2.

***Mathematica*. Синтаксис языка. Использование анонимных функций нескольких аргументов .**

В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в 3-й строке. Воспроизвести.

```
Clear[a,b,c];
???[a,b,c]//InputForm
a^3 - b + (b + c^3)^2
```

Отправить, что должно быть вместо ???.

Подсказки: При подготовке ответа не использовать # вместо #1. Пробелов в ответе быть не должно. В выражении ответа в Function[body] несколько аргументов, слагаемых. Чтобы не было разночтений в их расстановке, подготовленную часть ответа Function[body] уточните с использованием функций ToString, ToExpression (получите форму записи выражения, когда аргументы следуют, как в результате ToExpression). Число знаков проверяйте функцией StringLength. Символов в ответе - 20. Пробелов в ответе быть не должно

Рекомендуемая литература

1. *Воробьёв, Е. М.* Введение в систему “Математика” / Е. М. Воробьёв. - М. : Финансы и статистика, 1997. - 262 с.

2. *Таранчук, В.Б.* Основы работы с блокнотами Mathematica : учеб. материалы для студентов фак. прикладной математики и информатики / В. Б. Таранчук. – Минск : БГУ, 2015. – 52 с.

3. *Зеленица, А.М.* Mathematica. Виртуальный учебник. Перевод и оформление [Электронный ресурс]. URL : https://vk.com/doc-1172233_4590 62665?dl=304638db 29f87efde1.

4. WOLFRAM MATHEMATICA. Наиболее полная система для современных технических вычислений в мире [Электронный ресурс]. URL: http://www.wolfram.com/mathematica.

5. *Таранчук, В. Б.* Основные функции систем компьютерной алгебры : пособие для студентов фак. прикладной математики и информатики / В. Б. Таранчук. - Минск : БГУ, 2013. - 59 с.

6. WOLFRAM MATHEMATICA. Что нового в системе Mathematica 9 [Электронный ресурс]. URL: https://www.wolfram.com/mathematica/new-in-9/.

7. WOLFRAM MATHEMATICA. Новые функциональные возможности в Mathematica 10 [Электронный ресурс]. URL: http://www.wolfram.com /mathematica/new-in-10/.

8. WOLFRAM MATHEMATICA. Новое в системе Mathematica 11 [Электронный ресурс]. URL: http://www.wolfram.com/mathematica/new-in-11/.

9. WOLFRAM MATHEMATICA. Новое в системе Mathematica 12. Представленные категории Mathematica 12 [Электронный ресурс]. URL: http://www.wolfram.com/mathematica/new-in-12/.

10. WOLFRAM MATHEMATICA. Новое в системе Mathematica 13. Представленные категории [Электронный ресурс]. URL: http://www.wolfram.com/mathematica/new-in-13/.

Инструменты интерактивного программирования в системе *Mathematica*

Таранчук Валерий Борисович
БГУ,
факультет прикладной математики и информатики

Учебные материалы, инструкции и рекомендации пользователям системы компьютерной алгебры *Mathematica*, обучающие примеры и упражнения

Материалы темы “Манипуляции со списками” > > > >

- ✓ Функции формирования, редактирования списков
- ✓ Задачи серии “Список целых чисел”

Изложение тем “Wolfram Language. Списки”, “О формировании списков”, “Массивы с индексированными переменными”, “Списки. Примеры использования Count с levelspec”, “Выделение, удаление, дополнение элементов в списках”, “Манипуляции со списками”, “Шаблоны в списках” содержится в [1]. Напомним основные функции работы со списками, их формирования, обработки, манипулирования ими:

Функции формирования, редактирования списков

Функции выявления структуры списков:

- Count[list,pattern] – возвращает количество элементов в списке list, которые соответствуют образцу pattern;
- Dimensions[list] – возвращает размерность списка list (в формате списка размеров по каждому направлению);
- Length[list] – возвращает длину списка; для одномерного списка list возвращает число элементов, в случае многомерного списка – число размерностей;
- ArrayDepth – возвращает глубину вложенного списка; если вложенный список состоит из списков одинаковой длины, то для определения глубины этого вложенного списка, то есть, числа уровней, на которое необходимо спуститься, чтобы добраться до выражения, не являющегося списком, используется эта функция;
- VectorQ[list] – проверяет, является ли список вектором, дает True, если это так, и False в противном случае;
- MatrixQ[list] – проверяет, является ли список матрицей, и дает True, если это так, и False в противном случае;
- ArrayQ[list] – проверяет, является ли список массивом, и дает True, если это так, и False в противном случае;

- MemberQ[list,form] – проверяет, есть ли form в списке, и возвращает True, если это так и False в противном случае;
- Position[list,form] – возвращает номер позиции form в списке;
- TensorRank[list] – находит ранг списка, если он тензор.

Функции составления списков:

- Range[imin,imax,di] – возвращает список числовых элементов от наименьшего значения imin с приращением di до значения меньшего или равного imax (3 аргумента); эта функция применяется для создания ранжированных числовых элементов, значения которых лежат в задаваемом числовом диапазоне; могут быть 1 - 3 аргументов; числа могут быть целые, рациональные или вещественные;
- Range[imin,imax] – возвращает список целых чисел с заданными начальным и конечным значениями (если в обращении два аргумента);
- Range[imax] – возвращает список числовых элементов {1, 2, ..., imax};
- Table[expr,{imax}] – генерирует таблицу-список, содержащий imax экземпляров выражения expr (элементы могут быть вещественными и целыми числами или выражениями);
- Table[expr,{i,imax}] – генерирует список значений expr для i от 1 до imax;
- Table[expr,{i,imin,imax}] – начинает со значения i = imin;
- Table[expr,{i,imin,imax,di}] – использует шаги di;
- Table[expr,{i,imin,imax},{j,jmin,jmax},...] – возвращает вложенный список; самым внешним является список по переменной i; внешний итератор может зависеть от значения внутреннего итератора, в результате чего могут создаваться непрямоугольные списки – см. пример ниже, в котором количество столбцов в таблице меняется с изменением номера строки; внутренний итератор зависеть от внешнего не может, так как при генерации списка внутренний итератор должен иметь фиксированное значение, пока изменяется внешний.
- RandomInteger – создает случайные числа, векторы и матрицы с целыми элементами; RandomInteger[{imin, imax}] возвращает случайное целое число из заданного диапазона; RandomInteger[{imin,imax},n] возвращает список n случайных целых чисел из заданного диапазона; RandomInteger[{imin, imax},{n1,n2,...}] возвращает массив n1×n2×... случайных целых чисел из заданного диапазона;
- RandomReal[range,{n1,n2,...}] возвращает массив n1×n2×... случайных действительных чисел из заданного диапазона range;
- RandomSample – создает списки с заданным числом случайных действительных чисел в заданных их пределах.

Массивы с индексированными переменными:

К категории множественных данных относятся массивы с индексированными переменными, которые могут быть одномерными (один

список), двумерными и многомерными (вложенные списки). Одномерные массивы обычно называют векторами, двумерные – матрицами. *Mathematica* позволяет создавать с использованием функции Array многомерные массивы, число элементов в них ограничено лишь объемом памяти компьютера.

- Array[a,n] порождает список (массив) длины n с элементами a[i], где i=1,2,...,n.
- Array[f,{n_1, n_2, …}] – итератор, заданный в виде {n_1, n_2, …}, приводит к созданию вложенного списка.
- Array[f,{n_1, n_2, …}, {r_1, r_2, …}] создает список с использованием стартовых значений индексов r_i (по умолчанию равно единице).
- Array[a,dims,origin] создает список, в котором индексы итераторов dims изменяются, начиная со значения origin, по умолчанию равного единице.
- Array[a,dims,origin,name] создает выражение, в котором заголовок List всюду заменен заголовком name.
- SlotSequence (краткая форма – ##, встроенный символ языка) – представляет последовательность значений (последовательность позиций), возвращаемых чистой функцией.

Выделение, удаление, дополнение элементов в списках:

- Part[list,m]. Каждый элемент списка однозначно определяется своим номером. Для выделения заданного m-го элемента списка list используется функция Part[list,m] (в системе Mathematica первый элемент списка индексируется единицей); при m>0 отсчет номеров элементов идет с начала списка, а при m<0 – с его конца. Функцию Part также можно задать с использованием последовательности квадратных скобок; для выделения элементов списка list используются двойные квадратные скобки: list[[m]] – выделяет m-ый элемент списка list с его начала (если m<0 – с конца); list[[{m,k,...}]] – выделяет m-ый, k-ый и т.д. элементы списка.
- Select[list,crit] – выбирает все элементы ei списка list, для которых crit[ei] имеет значение True.
- Select[list,crit,m] – выбирает из первых m элементов, для которых crit[ei] есть True.

Дополнение, удаление элементов в списке:

- Append[list,element] – добавляет элемент в конец списка.
- Prepend[list,element] – добавляет элемент в начало списка.
- Insert[list,element,m] – вставляет элемент в позицию m (отсчет позиции ведется с начала листа, а если задано отрицательное m, то с конца). Данной функцией можно включать элемент в несколько позиций, указав каждую в фигурных скобках и оформив это указание также в фигурных скобках: вместо m. При таком включении необходимо учитывать позицию данного включаемого элемента с учетом расширения списка включением в

него предшествующих элементов.

- Delete[list,m] позволяет удалить из списка произвольный m-ый элемент без замещения его новым. Если m положительно, отсчет удаленного элемента идет с начала списка, а если m отрицательно, то с конца. Можно удалить несколько элементов списка, указав их каждый в фигурных скобках и оформив перечисление в фигурных скобках. Если элементом списка является список, то он фигурирует как один элемент. Можно удалять избранный элемент из элемента списка, указав в фигурных скобках вместо m номер элемента списка в общем списке и номер удаляемого элемента во внутреннем списке.

Повторяемость в списке:

- Tally возвращает список имеющихся в исходном списке объектов и их числа; первый элемент в каждой паре – объект из сгенерированного списка, а второй – сколько раз он повторяется в этом списке.
- Cases[{e1, e2, ...}, pattern] – возвращает список тех ei, которые соответствуют заданному шаблону (pattern).
- Cases[{e1, ...}, pattern->rhs] или Cases[{e1, ...}, pattern:>rhs] – возвращает список значений rhs, соответствующих тем ei, которые подходят под шаблон pattern.
- Cases[expr, pattern, levelspec] – возвращает список всех частей выражения expr на уровнях, указанных спецификацией levelspec, и которые соответствуют шаблону pattern.

Манипуляции со списками, изменение порядка элементов в списке:

- Partition. Список list разбить на не перекрывающиеся части с длиной n (подсписки) можно, используя функцию Partition[list,m]; в частности, так можно вектор перевести в матрицу. Если количество элементов в списке не делится нацело на m, то последние k (k<m) элементов удаляются. Partition[list,m,d] – как предшествующая функция возвращает разбиение списка, но с отступом d. При d<m подсписки перекрываются.
- Flatten[list] – выравнивает (превращает в одномерный) список по всем его уровням (эта функция уменьшает число уровней в списке).
- Flatten[list,m] – выравнивает список по его m-уровням.
- Flatten[list,m,h] – выравнивает с заголовком h по его m-уровням.
- FlattenAt[list,m] – выравнивает подсписок, если он оказывается m-ым элементом списка list. Если m отрицательно, позиция отсчитывается с конца (в отличие от Flatten понижает уровень у указанной части обрабатываемого списка).
- Sort[list] – сортирует и возвращает элементы списка list в каноническом порядке (цифры располагаются по величине, буквы – в алфавитном порядке).
- Sort[list, p] – сортирует согласно функции упорядочения p.

- Reverse[list] – возвращает список с обратным порядком расположения элементов.
- RotateLeft[list] – возвращает список после однократного циклического переноса влево.
- RotateLeft[list,n] – возвращает список после n-кратного циклического переноса влево.
- RotateRight[list] – возвращает список после однократного циклического переноса вправо.
- RotateRight[list,m] – возвращает список после m-кратного циклического поворота вправо.
- Transpose[list] – осуществляет транспозицию (смену строк и столбцов) для двумерного списка (в случае, когда список – прямоугольная матрица, функция возвращает транспонированную).
- Transpose[list,m] – осуществляет транспозицию m-мерного списка.

Комбинирование списков, работа с множествами:

- Complement[list,list1,list2,...] – возвращает список list с элементами, которые не содержатся ни в одном из списков list1, list2,...
- Intersection[list1,list2,...] – (пересечение множеств) возвращает упорядоченный список элементов, общих для всех списков listN.
- Join[list1,list2,...] – объединяет списки в единую цепочку (выполняет конкатенацию); Join может применяться на любом множестве выражений, имеющих один заголовок.
- Union[list1,list2,...] – удаляет повторяющиеся элементы списков и возвращает отсортированный список всех различающихся между собой элементов, принадлежащих любому из перечисленных в аргументах списков listN; функция обеспечивает теоретико-множественное объединение списков.
- Union[list] – возвращает отсортированный вариант списка list, в котором опущены все повторяющиеся элементы.

Примеры применения перечисленных выше функций приведены в [1]. В текущем модуле сформулированы избранные задачи для избранных. Источником большинства из приведенных заданий является электронный ресурс http://wolframcenter.ru/ (в настоящее время недоступен) с задачами семинаров Г.М. Фридмана "Базовый семинар по Wolfram *Mathematica*: решение задач". В тексте ниже упражнения и задания формулируются с предварительными пояснениями (с изложением и комментариями решения), задания для самоконтроля – они без ответов, но содержат подсказки.

Задания выполняются в LMS Moodle, что накладывает дополнительные ограничения на отправляемые на проверку ответы – должны быть в формате InputForm.

Задачи серии “Список целых чисел”

Упражнение 1. Дан список целых чисел. Узнать, сколько раз в этом списке встречается каждый элемент.

Для подготовки исходных данных используем RandomInteger:

```
rL1 = RandomInteger[10, 30]
```

```
{9, 3, 8, 10, 2, 8, 2, 3, 0, 3, 9, 3, 10, 5,
 5, 10, 1, 1, 9, 1, 2, 6, 6, 10, 8, 3, 4, 2, 2, 1}
```

Так как предполагается использование в следующих упражнениях и заданиях, зафиксируем сформированный набор

```
rL1 = {9, 3, 8, 10, 2, 8, 2, 3, 0, 3, 9, 3, 10,
   5, 5, 10, 1, 1, 9, 1, 2, 6, 6, 10, 8, 3, 4, 2, 2, 1};
```

Сколько раз в этом списке встречается каждый элемент, выводит функция Tally:

```
Tally@rL1
```

```
{{9, 3}, {3, 5}, {8, 3}, {10, 4},
 {2, 5}, {0, 1}, {5, 2}, {1, 4}, {6, 2}, {4, 1}}
```

В секции вывода (результате) первый элемент в каждой паре – объект из анализируемого списка, второй – сколько раз он повторяется.

Задание СЦЧ-1:

***Mathematica*. Синтаксис языка. Работа со списками. Использование функций подсчитать, случаи по образцу.**

Дан список целых чисел. Оставить только те числа, которые встречаются более 3 раз (определить, какие элементы в исходном наборе встречаются более 3 раз, в списке с результатом оставить только их).
В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в 3-й строке. Воспроизвести.

rL1 ={9,3,8,10,2,8,2,3,0,3,9,3,10,5,5,10,1,1,9,1,2,6,6,10,8,3,4,2,2,1};
???:>x]
{3,10,2,1}

Отправить, что должно быть вместо **???**.
Подсказки: Пробелов в ответе быть не должно. Символов в ответе - 30, есть #>3

Упражнение 2. Дан список целых чисел. Определить и вывести числа, которые в заданном списке встречаются более 3 раз. Использовать подготовленный набор исходных данных, функцию Alternatives.

Сколько раз в списке встречается каждый элемент, фиксирует функция Tally. Функция Alternatives в сочетании с Apply (в сокращенной записи @@) обеспечивает, от каких именно элементов следует избавиться (использован альтернативный шаблон). Решение:

```
Alternatives @@ Cases[Tally@rL1, {x_, _?(# > 3 &)} :> x]
```

```
3 | 10 | 2 | 1
```

В секции вывода (результате) вертикальная черта между элементами – аналог логического сложения, ИЛИ.

Задание СЦЧ-2:

***Mathematica*. Синтаксис языка. Работа со списками. Использование функций подсчитать, удалить случаи по образцу, варианты, применить, случаи по образцу.**

Дан список целых чисел. Оставить только те числа, которые встречаются более 3 раз (определить, какие элементы в списке встречаются более 3 раз, оставить все такие элементы непосредственно в самом списке результате, причем, убрать из него не отвечающие данному условию элементы).
В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в 3-й строке. Воспроизвести.

rL1 ={9,3,8,10,2,8,2,3,0,3,9,3,10,5,5,10,1,1,9,1,2,6,6,10,8,3,4,2,2,1};
???:>x]]
{3, 10, 2, 2, 3, 3, 3, 10, 10, 1, 1, 1, 2, 10, 3, 2, 2, 1}

Отправить, что должно быть вместо **???**.
Подсказки: Пробелов в ответе быть не должно. Символов в ответе - 60, есть: #<=3 ves@@Cas

Упражнение 3. Дан список случайным образом перемешанных 0 и 1. Определить и вывести список подсписков (последовательность) из подряд идущих нулей.

Сгенерируем список нулей и единиц:

```
rL2 = RandomInteger[1, 30]
```

```
{1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,
 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1}
```

Так как предполагается использование в следующих упражнениях и заданиях, зафиксируем сформированный набор

```
rL2 = {1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
   0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1};
```

Применим к сформированному списку функцию Split, которая разбивает список на подмножества из подряд расположенных одинаковых элементов:

```
Split@rL2
```

```
{{1}, {0}, {1}, {0}, {1}, {0, 0, 0},
 {1, 1}, {0}, {1}, {0, 0, 0, 0}, {1, 1}, {0},
 {1, 1}, {0, 0, 0, 0, 0}, {1, 1}, {0}, {1}}
```

Уточним применение функции Split, так как в упражнении требуется оставить только списки, состоящие из нулей:

```
Cases[Split@rL2, {0 ..}]
```

```
{{0}, {0}, {0, 0, 0}, {0}, {0, 0, 0, 0}, {0}, {0, 0, 0, 0, 0}, {0}}
```

Выше использована функция Cases, в которой применен соответствующий шаблон вида {p..} – называется “повторяющимся” и возвращает последовательность из одного и более объектов, каждый из которых совпадает с указанным p.

Задание СЦЧ-3:

***Mathematica*. Синтаксис языка. Работа со списками. Использование функций длина, разделить на части, случаи по образцу, повторяющийся.**
Дан случайный список, состоящий только из нулей и единиц. Определить последовательность из подряд идущих нулей, имеющую наибольшую длину, вывести значение длины.
В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в 3-й строке. Воспроизвести.

rL2 = {1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1};
???
5

Отправить, что должно быть вместо **???**.
Подсказки: Пробелов в ответе быть не должно. Символов в ответе - 35, есть th/@Cait@rL2

Задание СЦЧ-3d:

***Mathematica*. Синтаксис языка. Работа со списками. Использование функций длина, самый долгий, повторяющийся, отложенное правило, последовательность незаполненных мест, заменить всё.**
Дан случайный список, состоящий только из нулей и единиц. Определить последовательность из подряд идущих нулей, имеющую наибольшую длину, вывести значение длины.
В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в

3-й строке. Воспроизвести.

rL2 = {1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1};
???
5

Отправить, что должно быть вместо **???**.
Подсказки: Пробелов в ответе быть не должно. Ответ отправлять после применения к коду InputForm. Символов в ответе - 44, есть: x:Lon gth[{x}] }:>L rL2/.{

Упражнение 4. Дан список целых чисел. Вывести адреса, по которым в списке располагаются простые числа.

Сгенерируем список из 20 целых чисел для упражнений:

```
rL3 = RandomInteger[{1, 100}, 20]
```

```
{85, 50, 65, 36, 4, 31, 40, 26, 92,
 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
```

Так как предполагается использование в следующих упражнениях и заданиях, зафиксируем сформированный набор

```
rL3 = {85, 50, 65, 36, 4, 31, 40, 26,
   92, 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52};
```

Применим к сформированному списку функцию Position в сочетании с функцией - критерием PrimeQ, они определят адреса, по которым в списке располагаются простые числа:

```
rL3
Position[rL3, _?PrimeQ]
```

```
{85, 50, 65, 36, 4, 31, 40, 26, 92,
 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
```

```
{{6}, {10}, {12}, {15}, {17}, {19}}
```

Задание СЦЧ-4:

***Mathematica*. Синтаксис языка. Работа со списками. Использование функций преобразовать с учётом индекса, условный оператор, простое число?.**
Дан случайный список целых чисел. Умножить в нем каждое простое число на его порядковый номер в этом списке. В решении использовать функции, упомянутые выше. В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в 4-й строке. Воспроизвести.

rL3 = {85, 50, 65, 36, 4, 31, 40, 26, 92, 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
???
{85, 50, 65, 36, 4, 31, 40, 26, 92, 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
{85, 50, 65, 36, 4, 186, 40, 26, 92, 20, 22, 564, 18, 32, 705, 98, 1241, 25, 1121, 52}

Отправить, что должно быть вместо **???**.
Подсказки: Пробелов в ответе быть не должно. Ответ отправлять после применения к коду InputForm. Символов в ответе - 42, есть: #]&,rL3 ed[I meQ[#]

Упражнение 5. Дан список целых чисел. Вывести список с оформлением – простые числа дать синим цветом шрифтом Bold.

```
rL3 = {85, 50, 65, 36, 4, 31, 40, 26,
   92, 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52};
```

Применим к сформированному списку функцию MapAt (преобразовать элемент) в сочетании с Position (позиция по образцу) с заданием стиля Blue, Bold:

```
rL3
MapAt[Style[#, Blue, Bold] &, rL3, Position[rL3, _?PrimeQ]]
```

```
{85, 50, 65, 36, 4, 31, 40, 26, 92,
 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
```

```
{85, 50, 65, 36, 4, 31, 40, 26, 92,
 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
```

Задание СЦЧ-5:

***Mathematica*. Синтаксис языка. Работа со списками. Использование функций длина, условный оператор, таблица значений, часть, простое число?.**
Дан случайный список целых чисел. Умножить в нем каждое простое число на его порядковый номер в этом списке. В решении использовать функции, упомянутые выше. В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в 4-й строке. Воспроизвести.

rL3 = {85, 50, 65, 36, 4, 31, 40, 26, 92, 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
???
{85, 50, 65, 36, 4, 31, 40, 26, 92, 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
{85, 50, 65, 36, 4, 186, 40, 26, 92, 20, 22, 564, 18, 32, 705, 98, 1241, 25, 1121, 52}

Отправить, что должно быть вместо **???**.
Подсказки: Пробелов в ответе быть не должно. Ответ отправлять после применения к коду InputForm. Символов в ответе - 64, есть: le[I th[rL3 eQ[rL3 *=i

Упражнение 6. Для заданного простого числа определить его порядковый номер в последовательности всех простых чисел.

Напомним, что в материалах [1] были примеры использования функции Prime, которая ставит в соответствие номеру простого числа его значение. Примеры:

```
{Prime[1], Prime[7]}
Prime@Range@13
```

```
{2, 17}
```

```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41}
```

Чтобы выполнить обратное действие (для заданного простого числа определить его порядковый номер), придется каждый раз для очередного простого числа p решать уравнения вида Prime[n] == p относительно n. Сделать это без дополнительных настроек не просто. Например, имеем в ответе системы для функции Solve и для Reduce:

```
Solve[Prime@n == 7, n]
```

... Solve: Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information.

```
{{n → 4}}
```

```
Reduce[Prime@n == 7, n]
```

... Reduce: This system cannot be solved with the methods available to Reduce.

```
Reduce[Prime[n] == 7, n]
```

Чтобы не было предупреждений от системы, нужно указать область допустимых решений, что задано в условии – работаем с целыми числами, точнее, с натуральными. В функции Solve это позволяет задавать третий (необязательный) аргумент:

```
Solve[Prime@n == 7, n, Integers]
```

```
{{n → 4}}
```

Задание СЦЧ-6:

***Mathematica*. Синтаксис языка. Работа со списками. Использование функций решить уравнения, часть, заменить всё, отложенное правило, простое число?.**
Дан случайный список целых чисел. Умножить в нем каждое простое число на его порядковый номер в последовательности простых чисел. В решении использовать функции, упомянутые выше.
В строках следующего блока (текст жирным шрифтом) приведены копии из секций In и Out. Код первой и второй строк обеспечивает получение результата, приведенного в 4-й строке. Воспроизвести.

rL3 = {85, 50, 65, 36, 4, 31, 40, 26, 92, 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
???
{85, 50, 65, 36, 4, 31, 40, 26, 92, 2, 22, 47, 18, 32, 47, 98, 73, 25, 59, 52}
{85, 50, 65, 36, 4, 341, 40, 26, 92, 2, 22, 705, 18, 32, 705, 98, 1533, 25, 1003, 52}

Отправить, что должно быть вместо **???**.
Подсказки: Пробелов в ответе быть не должно. Ответ отправлять после применения к коду InputForm. Символов в ответе - 59, есть: rL3/.(p_) [[1]])*p meQ:>(n/.So

Рекомендуемая литература

1. *Таранчук, В.Б.* Основы программирования на языке Wolfram : учеб. материалы для студентов фак. прикладной математики и информатики спец. 1-31 03 04 «Информатика» / В.Б. Таранчук. - Минск : БГУ, 2015. - 49 с.

Содержание

Учебное издание

Таранчук Валерий Борисович

ИНСТРУМЕНТЫ ИНТЕРАКТИВНОГО ПРОГРАММИРОВАНИЯ В СИСТЕМЕ *MATHEMATICA*

Учебные материалы для студентов факультета прикладной математики и информатики

В авторской редакции

Ответственный за выпуск *В. Б. Таранчук*

Подписано в печать 30.12.2022. Формат 60×84/16. Бумага офсетная.
Усл. печ. л. 3,02. Уч.-изд. л. 2,94. Тираж 50 экз. Заказ

Белорусский государственный университет.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/270 от 03.04.2014.
Пр. Независимости, 4, 220030, Минск.

Отпечатано с оригинал-макета заказчика
на копировально-множительной технике
факультета прикладной математики и информатики
Белорусского государственного университета.
Пр. Независимости, 4, 220030, Минск.