Wolfram *Mathematica*
Интерактивные вычисления и визуализация
Таранчук Валерий Борисович

Учебные материалы, рекомендации пользователям СКА *Mathematica*, обучающие примеры и упражнения.
Оригинал документа создан и демонстрируется в NB, конспект предоставляется студентам в формате PDF,
учебные материалы размещены на соответствующей странице курса в LMS Moodle.

Уважаемые читатели. В сгруппированных секциях ниже (и везде далее в подобных) размещён материал для демонстраций и обсуждения на лекциях. Такой материал каждый может подготовить сам (например, по записям на лекции) или, найдя аналогичный в других электронных ресурсах, книгах.

▶ Содержание учебных материалов по темам лекций, заданиям практических занятий, имена файлов

Основные конструкции и операции языка Wolfram Language.

Списки. Манипуляции со списками. Шаблоны в списках

▼ Wolfram Language. Списки: общее о списках, функции выявления структуры

▼ Общее о списках

Как и всякая программная система, *Mathematica* предназначена для компьютерной обработки информации. На уровне своего языка программирования и общения с пользователем система *Mathematica* оперирует с тремя основными классами данных: численные, символьные, списки.

Наиболее общим видом сложных данных в системе *Mathematica* являются списки (lists). Их, как правило, относят к данным множественного типа.

Список – выражение системы *Mathematica*, имеющее заголовок List.

Списки представляют совокупность однотипных или разнотипных данных, сгруппированных с помощью фигурных скобок. С помощью списков можно задавать более привычные типы сложных данных, например: векторы (одномерные), матрицы (двумерные) и многомерные массивы. При этом каждая группа элементов многомерных списков выделяется своей парой фигурных скобок, а в целом список выделяется общими скобками.

```
FullForm[{a, b, c, d, e, f}]
FullForm[{{a, b, c}, {d, e, f}}]
```

Списки относятся к данным сложной структуры, и характеризуются размером, который представляет собой произведение числа элементов списков по каждому направлению. Число направлений называют размерностью. Например, одномерный список является вектором и характеризуется числом элементов по единственному направлению. При этом вектор может быть

вектором-строкой и вектором-столбцом. Двумерный список представляет матрицу, имеющую m строк и n столбцов, её размер m×n. При работе со списками возникает необходимость контроля их структуры.

▼ Функции выявления структуры списков

Для выявления структуры списков используется ряд функций, перечислим основные:

- Count[list,pattern] – возвращает количество элементов в списке list, которые соответствуют образцу pattern;
- Dimensions[list] – возвращает список размеров списка list 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.

▼ Примеры использования Count, Dimensions, Length

▶ ✓ Count[list,pattern] – возвращает количество элементов в списке list, которые соответствуют образцу pattern

▶ ✓ Dimensions[list] – возвращает список размеров списка list 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

О формировании списков

Общее о составлении списков

Списки можно составлять напрямую, задавая объекты в соответствии с описанным выше синтаксисом. При этом элементы могут быть самой разной природы.

Примеры:

```
List[{a1, a + b x + c x^2, d, f^2, 12.3, 12345, 12 / 123}]
List[{a11, a12, a13}, {a21, a22, a23}] // MatrixForm
```

Функции составления списков

Можно генерировать различные виды списков, используя встроенные функции:

- 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.

▼ Примеры использования Range

▶ ✓ Range[imax] – возвращает список числовых элементов {1, 2, ..., imax}
✓ Range[imin,imax] – возвращает список целых чисел с заданными начальным и конечным значениями (если в обращении два аргумента);

▶ ✓ 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

▼ Примеры использования Table и TableForm

▶ ✓ 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 ...

▶ ✓ Table – табулирование функций.

▼ Списки. Примеры использования Count с levelspec

▶ ✓ Count[list,pattern] – возвращает количество элементов в списке list, которые соответствуют образцу pattern

▶ ✓ Count[list,pattern]. Примеры, сколько элементов категорий Power, Rational:

▶ ✓ Count[list,pattern]. Примеры, сколько элементов категорий Rational, Integer:

▶ ✓ Count[list,pattern]. Примеры, сколько элементов категорий x_^2:

▼ ✓ Count[list,pattern,levelspec] – возвращает количество элементов в списке list, которые соответствуют образцу pattern причем на указанном уровне levelspec

In[14]:=
```
Count[{a, b^2, {{a, b^2, c}, {d, e, f^2}}}, x_^2]
```

In[15]:=
```
(* Для понимания пользуйтесь *)TreeForm[{a, b^2, {{a, b^2, c}, {d, e, f^2}}}]
```

In[19]:=
```
Count[{a, b^2, {{a, b^2, c}, {d, e, f^2}}}, x_^2, 1]
```

In[18]:=
```
Count[{a, b^2, {{a, b^2, c}, {d, e, f^2}}}, x_^2, 2]
```

In[17]:=
```
Count[{a, b^2, {{a, b^2, c}, {d, e, f^2}}}, x_^2, 3]
```

▶ ✓ Count[list,pattern,levelspec] – ... на указанном уровне levelspec – на всех до levelspec

▶ ✓ Count[list,pattern,levelspec] – на указанном уровне levelspec – конкретно на levelspec

▶ ✓ Пример, как вывести то, что выдается в Position

```
Count[{a, b^2, c, d, e, f^2, 3^g}, _Power]
Count[{a, b^2, c, d, e, f^2, 3 / 2}, _Rational]
```

О формировании списков (*дополнительно – Range, Table, TableForm, Subdivide, Random**)

Примеры использования Range:

✓ Range[imin,imax,di] – возвращает список числовых элементов от наименьшего значения imin с приращением di до значения меньшего или равного imax (3 аргумента); ...

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[1.23, 2.34, 0.11]
Range[a, b, (b - a) / 5]
```

✓ Примеры использования Subdivide: Subdivide – разбить интервал на равные части ...

Примеры использования Table и TableForm:

✓ 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.

```
Table[10 i + j, {i, 3}, {j, 3, 700, 100}] // MatrixForm
```

```
Table[10 i + j, {i, 3}, {j, 3, 700, 100}] // TableForm
```

Примеры использования Random*, Range:

- 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.

Примеры формирования списков ...:

▶ ... список N чисел, используя RandomInteger, с заданием интервала K - L:

▶ ... список N чисел, используя Range, и перемешиваем их:

▶ ... случайный список, состоящий только из нулей и единиц:

▶ ... случайный список, состоящий только из -1, 0:

▶ ... список всех двузначных целых чисел:

▶ ... список простых чисел от 1 до 100:

▶ ... список целых чисел от 1 до 100, которые делятся на 7, вариант Range:

▼ ... список целых чисел от 1 до 100, которые делятся на 7, вариант Cases:

```
Cases[Range[100], x_ /; Divisible[x, 7]]
```

Примеры формирования матриц, ...:

▶ ... матрица m×n с элементами: целые из заданного интервала:

▶ ... матрица 5×3 со строками, содержащими номер строки и два случайных числа:

▼ Выделение, удаление, дополнение элементов в списках

▼ Выделение (извлечение) элементов

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.

Примеры:

▶ ✓ Возьмем из создаваемого списка третий элемент (три варианта записи):

▶ ✓ Возьмем из созданного списка второй элемент с конца (три варианта записи):

▶ ✓ Примеры – диапазон, из интервала, с заданным шагом:

Извлечение возможно для списков любой степени вложенности, требуется задать уровень. Ниже примеры извлечения третьего элемента списка, в третьем элементе второго вложенного элемента, всех первых элементов всех вложенных списков:

▶ ✓ Пример извлечения третьего элемента списка и в третьем элементе второго вложенного элемента :

▶ ✓ Пример извлечения всех первых (вторых) элементов всех вложенных списков

✓ Пример использования ## и FromDigits в сочетании с &

▼ Извлечение элементов по признакам (использование Select)

Для выделения элементов списка, удовлетворяющих заданному критерию применяются:

- **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.

Ниже отдельно приведены примеры работы с функциями Cases, DeleteCases.

▶ ✓ В примерах ниже в списке vectr выбираются четные и превышающие 55:

В примерах ниже в списке list1 из пар выбираются только те, где на втором месте присутствует b и, где на первом – четное число:

▶ ✓ .. где на втором месте присутствует b:

► ✓ .. где на первом – четное число:

Дополнение элементов в списке

Для расширения списка путем включения в него новых элементов, используются следующие функции:

- 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). Можно данной функцией 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.

Примеры ниже – добавление в конец списка, в начало, в указываемые места, в том числе с отсчетом позиции вставки от начала и от конца списка:

```
vectr = Range[11, 99, 11]
Append[vectr, 123]
Prepend[vectr, 1]
Insert[vectr, 222, 3]
Insert[vectr, 777, -3]
```

Пример ниже – добавление на указанные позиции (оригинала, а не с учётом добавляемых):

```
Insert[vectr, ab, {{2}, {4}, {-2}}]
```

Удаление элементов из списка

Функция 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.

Примеры формирования списка, удаления указанных элементов:

```
vectr = Range[11, 99, 11]
Delete[vectr, 2]
Delete[vectr, -2]
```

▶ ✓ Пример ниже – обратите внимание, как надо перечислять удаляемые элементы:

В списке list2 3-й элемент – список; 3-я строка секции ниже – пример, как во внутреннем списке удалить конкретные элементы:

In[152]:=

```
list2 = {11, 22, {33, 331, 332}, 44, 55, 66, 77, 88}
Delete[list2, 3]
Delete[list2, {{3}, {5}}]
```

▶ ✓ .. пример, как во внутреннем списке удалить конкретные элементы:

Повторяемость в списке

Функция **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:

▶ ✓ Пример ниже – сформировали, используя RandomInteger, вывели сколько раз встречается ...:

- **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

▶ ✓ .. сформировали, вывели сколько раз встречается .., отметили, какие более 3-х раз:

Манипуляции со списками

Разделение списка на части

Список list разбить на неперекрывающиеся 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. Примеры ниже:

```
vectr = Range[11, 99, 11]
Partition[vectr, 3] // MatrixForm
Partition[vectr, 3, 2] // MatrixForm (* с отступом *)
```

Изменение порядка элементов в списке

Кроме дополнения списков новыми данными имеется возможность изменения порядка расположения элементов в списке:

- 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.

Пример ниже – как превратить список в одномерный:

In[157]:=

```
list3 = {11, {22, {221, {2211, 2212}}}, {33, 331, 332}, 44}
Flatten[list3]
```

Примеры FlattenAt[list,m] – выравнивает (превращает в одномерный) подсписок, если он оказывается m-ым элементом списка list:

In[159]:=

```
list3
FlattenAt[list3, 2]
FlattenAt[list3, 3]
```

Примеры Sort[list] – сортирует и возвращает элементы списка list, цифры располагаются по величине, буквы – в алфавитном порядке:

In[162]:=

```
list4 = {11, 22, 221, 222, ab223, 33, 331, cd332, 44}
Sort[list4]
```

Примеры Sort[list, p] – сортирует согласно функции упорядочения p (Greater – больше чем, Less – меньше чем):

```
list5 = {11, 22, 221, 222, 33, 331, 44}
Sort[list5, Greater]
Sort[list5, Less]
```

Пример сортировки с определением функции-правила (#1>#2&, действует для цифр):

```
Sort[list5, #1 > #2 &]
```

Примеры Reverse[list] – возвращает список с обратным порядком расположения элементов:

```
list4
Reverse[list4]
list3
Reverse[list3]
```

RotateLeft[list] – возвращает список после однократного циклического переноса влево; RotateLeft[list,n] – возвращает список после n-кратного циклического переноса влево:

```
list4
RotateLeft[list4]
RotateLeft[list4, 2]
```

Формируем матрицу, показываем Transpose[list] – в случае, когда список прямоугольная матрица, осуществляет транспозицию

(смену строк и столбцов):

In[164]:=
```
m = Table[10 i + j, {i, 4}, {j, 3}]
{MatrixForm[m], MatrixForm[Transpose[m]]}
```

Комбинирование списков и работа с множествами

При необходимости комбинирования нескольких списков можно использовать следующие функции:

- 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 listi;
- 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 listi; функция обеспечивает теоретико-множественное объединение списков;
- Union[list] – возвращает отсортированный вариант списка list, в котором опущены все повторяющиеся элементы.

Подготовка списков для упражнений:

```
vectr = Range[11, 99, 11]
vectr2 = Insert[vectr, 777, -3]
vectr3 = Delete[vectr, -3]
```

Примеры работы с Complement[list,list1,list2,...] – возвращает список list с элементами, которые не содержатся ни в одном из списков list1, list2, ...:

```
vectr
vectr2
Complement[vectr, vectr2] (* уникального во 2-ом нет *)
Complement[vectr2, vectr]
```

```
vectr
vectr3
Complement[vectr, vectr3]
```

Примеры работы с Intersection[list1,list2,...] – возвращает упорядоченный список элементов, общих для всех списков listi:

```
vectr
vectr2
vectr3
Intersection[vectr, vectr2]
Intersection[vectr2, vectr3]
```

Примеры работы с Join[list1,list2,...] – объединяет списки в единую цепочку; Union[list1,list2,...] – удаляет повторяющиеся элементы списков и возвращает отсортированный список всех различающихся между собой элементов, принадлежащих любому из перечисленных в аргументах списков listi:

```
vectr
vectr2
vectr3
Join[vectr2, vectr3]
Union[%]
Union[vectr2, vectr3]
Union[vectr, vectr2]
```

Замечание. Далее отдельно рассматриваются возможности работы со списками в стеке, специальные функции обработки списков Tally, Riffle, Split, функции и опции вывода элементов списков.

Шаблоны в списках

Функция Cases

Эффективным инструментом манипулирования со списками являются шаблоны. Функция **Cases**[{e1,e2,...},pattern] возвращает список тех eN, которые соответствуют заданному шаблону pattern.

Примеры выбора элементов, отвечающих условиям

Можно создавать шаблоны любой сложности, обрабатывать любые уровни. Несколько примеров с указанием правил, выбор только списков, списков с нечетным первым элементом (уровни неявно уточняются, т.к. выбираем пары, т.е. список):

✓ .. выбор только списков с двумя элементами:

```
list6 = {1, 2.3, 3, 2, 3.4, {3, 3.3}, {one, three}, 2, three, {7, three}, {8, "one"}, {9, one, three}}
```

In[39]:=

```
list6
Cases[list6, {p_, q_}] (* выбрать все, где пары *)
```

▶ ✓ .. выбор только списков с тремя элементами:

▶ ✓ .. выбор только списков пар и с нечетным первым элементом:

▼ Примеры с указанием уровней

Примеры выбора: только списков, списков с нечетным первым элементом, списков со вторым элементом, квадрат которого больше заданного значения:

In[4]:=

```
list5 = {1, 2.3, 3, 2, 3.4, {3, 3.3}, {one, three}, 2, three, {7, three}, {8, one}}
Cases[list5, _List]
```

▶ ✓ .. списков с нечетным первым элементом, но только в парах:

Замечание. Эффективной конструкцией составления шаблонов является Condition – сокращенная форма записи /;

▶ ✓ .. с первым элементом, квадрат которого больше заданного значения:

▶ ✓ .. из пар с первым элементом, квадрат которого больше заданного значения:

▶ ✓ .. из пар со вторым элементом, квадрат которого больше заданного значения:

▶ ✓ .. по всем с .., на уровне 2:

▼ Функции удаления по шаблонам

Дополнительно отметим функции удаления по шаблонам:

- DeleteCases[expr,pattern] – исключает все элементы выражения expr, которые совпадают с образцом pattern;

- DeleteCases[expr,pattern,levspec] – исключает все части выражения expr на уровнях, указанных levspec, и соответствующих образцов pattern (конкретная глубина поиска задается аргументом в виде списка).

  Примеры:

  ▶ ✓ .. удаление, где на 1-ом уровне целые:

  ▶ ✓ .. удаление, где на 2-ом уровне целые:

  ▶ ✓ .. удаление, где во вложенных списках пары:

  ▶ ✓ .. удаление, где на базовом уровне элемент, квадрат которого больше заданного значения:

  ▶ ✓ .. удаление, где на вложенном уровне элемент, квадрат которого больше заданного значения:

▶ Массивы с индексированными переменными

▶ Массивы с индексированными переменными *(повторно)*.

▶ Рекомендуемая литература