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

▼ План лекции

О языке Wolfram Language (*Mathematica*)
Повторение пройденного: Части выражений. Основные функции
Выделение, удаление, дополнение элементов в списках
Манипуляции со списками
Шаблоны в списках
Массивы с индексированными переменными

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

О формировании списков

Списки. Примеры использования Count с levelspec

▼ *Повторение пройденного*:

Задание. Внимательно просмотрите в конспекте:

Основные формы выражений

▼ Части выражений. Основные функции

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.

✓ Head[expr] – возвращает тип, нулевая часть (0-ая часть);

✓ Part[expr,m] выделяет m-ую часть выражения (или списка) с начала; если m отрицательно, позиция отсчитывается с конца, эквивалентные записи: expr[[m]] (две подряд квадратные скобки), expr⟦…⟧ (двойные квадратные скобки);

✓ Part[expr,n_1;;n_2] – части выражения (или списка) от n_1 до n_2;

✓ First[expr] – возвращает первый элемент в выражении (или в списке);

✓ Last[expr] – возвращает последний элемент в выражении (или в списке);

✓ Numerator[expr] – возвращает числитель выражения или указанной части;

✓ Denominator[expr] – возвращает знаменатель выражения или указанной части.

Части выражений. Примеры

Части выражений, формы записи. Примеры

Задание. Прочитайте общее описание функции **позиция по образцу**, составьте и выполните собственные упражнения вариантов:

✓ Find all positions at which x to any power appears,

✓ Find only those down to level 2,

✓ Use Extract to extract parts based on results from Position:

Задание. Составьте собственные упражнения типа приведенных на лекции:

✓ Numerator[expr] – возвращает числитель выражения или указанной части (код доступа 04181023):

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, задачи для самостоятельного выполнения:

▶ ✓ 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

▼ 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:

```
vectr = Range[11, 99, 11]
Select[vectr, EvenQ]
Select[vectr, # > 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]
```

▶ ✓ Как не работает, 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:

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

Задание. Прочитайте общее описание функции **удалить элемент**, составьте и выполните собственные упражнения вариантов:
✓ Delete in a 2D array,
✓ Delete at positions 1 and 3,
✓ Delete at any position in any expression

▶ ✓ .. пример, **как во внутреннем списке** удалить конкретные элементы (код доступа 04181110):

▼ 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 ...:

```
randomInt = RandomInteger[{7, 25}, 20]
Sort[%, Less]
Tally@randomInt
```

■ **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:

```
randomInt2 = RandomInteger[{5, 11}, 20]
Tally@randomInt2
Cases[Tally@randomInt2, {x_, _?(# > 3 &)} :> x]
```

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:

```
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:

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

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

```
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):

```
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}}
```

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

✓ .. выбор только списков с тремя элементами:

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

```
list6
Cases[list6, {_?OddQ, _}](* выбрать все, где пары, причем, на первом месте нечётные *)
```

▼ Примеры с указанием уровней

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

```
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:

```
list5
Cases[list5, {_?OddQ, q_}]
list6
Cases[list6, {_?OddQ, q_}]
```

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

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

```
list5
Cases[list5, q_ /; q^2 > 5] (*уровень не указываем, по умолчанию - первый *)
```

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

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

```
list5
Cases[list5, {p_, q_ /; q^2 > 5}]
```

▶ ✓ .. 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 и задачи для самостоятельного выполнения:

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

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

```
list9
DeleteCases[list9, _Integer, {2}]
```

✓ .. удаление, где во вложенных списках пары:

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

```
list9
DeleteCases[list9, q_ /; q^2 > 5]
```

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

```
list9
DeleteCases[list9, q_ /; q^2 > 5, {2}]
```

Массивы с индексированными переменными

Также 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.

Функция Array[a,n] порождает список длины n с элементами a[i], где i=1,2,...,n. Итератор, заданный в виде {n1,n2,...}, приводит к созданию вложенного списка (Array[f, {n1, n2, ...}]). Функция Array[a,dims,origin] дает список, в котором индексы итераторов dims изменяются, начиная со значения origin, по умолчанию равного единице. Если же функция задана в полной форме, Array[a,dims,origin,name], то получается выражение, в котором заголовок List всюду заменен заголовком name.

Примеры задания массивов и их вывод:

```
Array[a, 7]
Array[a, 7, 3]
```

```
Array[#^3 &, 7, 5]
```

```
Array[Sin, 3, π / 4]
Array[Sin, 3, {0, 1}]
Array[Sin, 3, π / 4, Plus]
```

```
Array[f, 8, {0, 1}]
```

```
Array[b, {3, 5}]
Array[10 #1 + #2 &, {3, 5}]
Array[a### &, {3, 5}]
```

SlotSequence (краткая форма – ##, встроенный символ языка) – представляет последовательность значений (последовательность позиций), возвращаемых чистой функцией. Примеры:

```
Array[## &, {1, 1}]
Array[## &, {1, 2}]
```

```
Array[## &, {1, 5}]
```

```
Array[### &, {5, 1}]
```

```
Array[##& , {5, 2}]
```

```
Array[##& , {5, 2}] // MatrixForm
```

```
{Array[#& , {3, 5}] // MatrixForm, Array[##& , {3, 5}] // MatrixForm}
```

Дополнительные примеры задания массивов и их вывод (ранее рассмотрели с Table):

```
Table[10 i + j, {i, 3}, {j, 4}] // MatrixForm
Array[FromDigits[{##}] &, {3, 4}] // MatrixForm
```

Массивы с индексированными переменными (*задачи для самостоятельного выполнения*).

✓ ... Функция Array[a,dims,origin] дает список, в котором индексы итераторов dims изменяются, начиная со значения origin, по умолчанию равного единице ...

Обратить внимание, что можно вместо #1 писать #, но в тестовых заданиях отдельно оговорено ... Пояснить почему

```
Array[10 #1 + #2 &, {3, 5}] // MatrixForm
Array[10 # + #2 &, {3, 5}] // MatrixForm
```

✓ SlotSequence (краткая форма – ##, встроенный символ языка) – представляет последовательность значений (последовательность позиций), возвращаемых чистой функцией

✓ Примеры использования SlotSequence (краткая форма – ##, встроенный символ языка)

✓ Пример сравнения # & и ## &

✓ Примеры FromDigits, Array[FromDigits[{##}]

✓ Примеры FromDigits, Array[FromDigits[{##}], “расшифруем” что делает

✓ Пример использования ## и FromDigits в сочетании с &