Компьютерная математика. Символьный пакет Mathematica

Часть 2

Тема 6. Система “Аналитическая геометрия на плоскости”. Взаимное расположение геометрических объектов Точка, Вектор, Прямая

Лаврова Ольга Анатольевна

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ММФ, кафедра дифференциальных уравнений и системного анализа (ауд. 341)

Система “Аналитическая геометрия на плоскости”

Система “Аналитическая геометрия на плоскости” разработана в рамках дисциплины Компьютерная математика. Часть1.

В системе “Аналитическая геометрия на плоскости” определены три типа геометрических объекта: Точка, Вектор, Прямая.

Для каждого типа геометрических объектов определены

1. **внутреннее представление объекта --** структура данных для хранения информации об объекте
2. **функции-конструкторы** для создания геометрических объектов
3. **функции-свойства** для извлечения свойств геометрических объектов
4. **графический образ --** способ отображения геометрического объекта с помощью графических примитивов.

Внутреннее представление геометрических объектов

Внутренним представление геометрического объекта является выражение со строго определенной головой и аргументами, соответствующими этой голове.

Геометрический объект типа Точка

Внутренним представлением геометрического объекта типа Точка является выражение вида

```
In[1]:= kmPoint[km, id_String, {x_, y_}];
```

- Голова **kmPoint** является строго определенной для геометрического объекта типа Точка
- Первый аргумент **km** является маркером того, что выражение является внутренним представлением некоторого геометрического объекта системы “Аналитическая геометрия на плоскости”
- Второй аргумент **id_String** содержит имя геометрического объекта в виде строкового выражения. Для создания объекта без имени используется пустая строка “” в качестве значения второго аргумента.
- Третий аргумент **{x_,y_}** определяет координаты геометрического объекта типа Точка в декартовой прямоугольной системе координат.

Геометрический объект типа Вектор

Внутренним представлением геометрического объекта типа Вектор является выражение вида

In[2]:=
```
kmVector[km, id_String, {x_, y_}];
```

- Голова **kmVector** является строго определенной для геометрического объекта типа Вектор
- Первый аргумент **km** является маркером того, что выражение является внутренним представлением некоторого объекта системы “Аналитическая геометрия на плоскости”
- Второй аргумент **id_String** содержит имя геометрического объекта в виде строкового выражения. Для создания объекта без имени используется пустая строка “” в качестве значения второго аргумента.
- Третий аргумент **{x_,y_}** определяет координаты геометрического объекта типа Вектор в декартовой прямоугольной системе координат.

Геометрический объект типа Прямая

Внутренним представлением геометрического объекта Прямая является выражение вида

In[3]:=
```
kmLine[km, id_String, {A_, B_, C_}];
```

- Голова **kmLine** является строго определенной для геометрического объекта типа Прямая
- Первый аргумент **km** является маркером того, что выражение является внутренним представлением некоторого объекта системы “Аналитическая геометрия на плоскости”
- Второй аргумент **id_String** содержит имя геометрического объекта в виде строкового выражения. Для создания объекта без имени используется пустая строка “” в качестве значения второго аргумента.
- Третий аргумент **{A_,B_,C_}** определяет коэффициенты в общем уравнении прямой на плоскости $A x + B y + C == 0$.

Функции-конструкторы для создания геометрических объектов

Важное соглашение для имен функций-конструкторов: имя функции-конструктора для создания геометрического объекта совпадает с головой внутреннего представления создаваемого объекта (kmPoint, kmVector, kmLine).

Функции-конструкторы делятся на два класса: функции-конструкторы для создания геометрических объектов с именем и функции-конструкторы для создания геометрических объектов без имени.

Геометрический объект типа Точка

Для создания геометрического объекта типа Точка с именем определены **две функции-конструктора**.

1. Одна функция-конструктор создает объект типа Точка по координатам в декартовой прямоугольной системе координат

```
In[4]:= kmPoint[id_String, {x_, y_}] := kmPoint[km, id, {x, y}]
```

2. Вторая функция-конструктор создает объект по координатам точки в полярной системе координат

```
In[5]:= kmPoint[id_String, {ρ_, ϕ_}, "pol"] := kmPoint[km, id, {ρ Cos[ϕ], ρ Sin[ϕ]}]
```

```
In[6]:= ? kmPoint
```

```
Global`kmPoint

kmPoint[id_String, {x_, y_}] := kmPoint[km, id, {x, y}]

kmPoint[id_String, {ρ_, ϕ_}, pol] := kmPoint[km, id, {ρ Cos[ϕ], ρ Sin[ϕ]}]
```

Для создания геометрического объекта типа Точка без имени определены две аналогичные функции-конструктора.

Геометрический объект типа Вектор

Для создания геометрического объекта типа Вектор с именем определены **три функции-конструктора.**

1. Одна функция-конструктор создает объект типа Вектор по координатам вектора в декартовой прямоугольной системе координат

```
In[7]:= kmVector[id_String, {x_, y_}] := kmVector[km, id, {x, y}]
```

Данная функция-конструктор определена как **базовый конструктор**. Это означает, что остальные функции-конструкторы для создания объекта типа Вектор нужно строить на основании базового конструктора.

2. Вторая функция-конструктор создает объект типа Вектор по координатам начальной и конечной Точек вектора

```
In[8]:= kmVector[id_String, P0_kmPoint, P1_kmPoint] := kmVector[id, P1@"coord" - P0@"coord"]
```

Данный конструктор строится на основании базового конструктора.

Определим дополнительно функцию-свойство для геометрического объекта типа Точка с идентификатором “coord”, которая возвращает декартовы координаты объекта типа Точка

```
In[9]:= kmPoint[_, _, coords_]["coord"] := coords
```

3. Третья функция-конструктор создает объект типа Вектор по заданному орту (Вектору единичной длины) и длине вектора

```
In[10]:= kmVector[id_String, ort : kmVector[km, _, coord_], length_] := kmVector[id, length * coord]
```

Обратите внимание, что конструктор строится на основании базового конструктора.

In[11]:=
```
? kmVector
```

```
Global`kmVector
```

```
kmVector[id_String, {x_, y_}] := kmVector[km, id, {x, y}]

kmVector[id_String, P0_kmPoint, P1_kmPoint] := kmVector[id, P1[coord] - P0[coord]]

kmVector[id_String, ort : kmVector[km, _, coord_], length_] := kmVector[id, length coord]
```

Тестируем созданные конструкторы

In[12]:=
```
V1 = kmVector["V1", {0, 1}]
```

Out[12]=
```
kmVector[km, V1, {0, 1}]
```

In[13]:=
```
P1 = kmPoint["P1", {0, 1}]; P2 = kmPoint["P2", {0, 2}];
```

In[14]:=
```
V2  =  kmVector["V2", P1, P2]
```

Out[14]=
```
kmVector[km, V2, {0, 1}]
```

In[15]:=
```
V3  =  kmVector["V3", V1, 5]
```

Out[15]=
```
kmVector[km, V3, {0, 5}]
```

Для геометрического объекта типа Вектор без имени определены три аналогичные функции-конструктора.

Геометрический объект типа Прямая

Для геометрического объекта типа Прямая с именем определены **четыре вида функций-конструкторов.**

1. Одна функция-конструктор создает объект типа Прямая по общему уравнению прямой

In[16]:=
```
kmLine[id_String, A_. * x_ + B_. * y_ + C_. == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {A, B, C}]
```

In[17]:=
```
kmLine[id_String, B_. * y_ + C_. == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {0, B, C}]
```

In[18]:=
```
kmLine[id_String, A_. * x_ + C_. == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {A, 0, C}]
```

2. Вторая функция-конструктор создает геометрический объект типа Прямая по заданному объекту типа Точка на прямой и направляющему Вектору

In[19]:=
```
kmLine[id_String, P0_kmPoint, Dir_kmVector] := kmLine[id,
  Det[{Dir@"coord", {x, y} - P0@"coord"}] == 0, {x, y}]
```

Данная функция-конструктор определена как **базовый конструктор**. Это означает, что остальные конструкторы для создания геометрического объекта типа Прямая будут сторится на основании базового конструктора.

Определим дополнительно функцию-свойство для геометрического объекта Вектор с идентификатором “coord”, которая возвращает декартовы координаты вектора

In[20]:=
```
kmVector[_, _, coords_]["coord"] := coords
```

3. Третья функция-конструктор создает объект типа Прямая по двум Точкам, лежащим на прямой

In[21]:=
```
kmLine[id_String, P0_kmPoint, P1_kmPoint] := kmLine[id, P0, kmVector["dir", P1@"coord" - P0@"coord"]]
```

Обратите внимание, что конструктор строится на основании базового конструктора по точке и направляющему вектору.

4. Четвертая функция-конструктор создает объект типа Прямая по Точке на прямой и нормальному Вектору

In[22]:=
```
kmLine[id_String, P0_kmPoint, {"normal", norm_kmVector}] := "Should be specified"
```

Конструктор необходимо построить на основании базового конструктора.

In[23]:=
```
? kmLine
```

```
Global`kmLine

kmLine[id_String, C_. + A_. x_ + B_. y_ == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {A, B, C}]

kmLine[id_String, C_. + B_. y_ == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {0, B, C}]

kmLine[id_String, C_. + A_. x_ == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {A, 0, C}]

kmLine[id_String, P0_kmPoint, Dir_kmVector] := kmLine[id, Det[{Dir[coord], {x, y} - P0[coord]}] == 0, {x, y}]

kmLine[id_String, P0_kmPoint, P1_kmPoint] := kmLine[id, P0, kmVector[dir, P1[coord] - P0[coord]]]

kmLine[id_String, P0_kmPoint, {normal, norm_kmVector}] := Should be specified
```

Тестируем созданные конструкторы

In[24]:=
```
L1 = kmLine["Line1", y - x + 5 == 0, {x, y}]
```

Out[24]=
```
kmLine[km, Line1, {-1, 1, 5}]
```

```
In[25]:= P1 = kmPoint["P1Name", {0, 0}]; P2 = kmPoint["P2Name", {1, 1}];

In[26]:= V1 = kmVector["V1", {1, 1}]
Out[26]= kmVector[km, V1, {1, 1}]

In[27]:= L2 = kmLine["Line2", P1, V1]
Out[27]= kmLine[km, Line2, {-1, 1, 0}]

In[28]:= L3 = kmLine["Line3", P1, P2]
Out[28]= kmLine[km, Line3, {-1, 1, 0}]

In[29]:= L4 = kmLine["Line4", P1, {"normal", V1}]
Out[29]= Should be specified
```

Для геометрического объекта типа Прямая без имени определены четыре аналогичные функции-конструктора.

Свойства геометрических объектов

Для извлечения свойств геометрических объектов используются функции-свойства. Левая часть функции-свойства имеет вид

```
In[30]:= kmPoint[km, __]["propertyName"];

In[31]:= kmVector[km, __]["propertyName"];

In[32]:= kmLine[km, __]["propertyName"];
```

“propertyName” является идентификатором запрашиваемого свойства.

Рассмотрим идентификаторы свойств, определенные в системе “Аналитическая геометрия на плоскости”

1. Функция-свойство с идентификатором “**coord**” возвращает декартовы координаты соответствующего геометрического объекта типа Точка или типа Вектора.

Функции-свойства с идентификатором “coord” для геометрических объектов Точка и Вектор были определны выше

```
In[33]:= ? kmPoint
```

```
Global`kmPoint

kmPoint[_, _, coords_][coord] := coords

kmPoint[id_String, {x_, y_}] := kmPoint[km, id, {x, y}]

kmPoint[id_String, {ρ_, ϕ_}, pol] := kmPoint[km, id, {ρ Cos[ϕ], ρ Sin[ϕ]}]
```

In[34]:=
```
? kmVector
```

```
Global`kmVector

kmVector[_, _, coords_][coord] := coords

kmVector[id_String, {x_, y_}] := kmVector[km, id, {x, y}]

kmVector[id_String, P0_kmPoint, P1_kmPoint] := kmVector[id, P1[coord] - P0[coord]]

kmVector[id_String, ort : kmVector[km, _, coord_], length_] := kmVector[id, length coord]
```

2. Функция-свойство с идентификатором “**id**” возвращает имя соответствующего геометрического объекта любого из типов

Определим, что функция-свойство с идентификатором “id” возвращает имя соответствующего геометрического объекта, если оно определено, либо имя по умолчанию. Именами по умолчанию являются “Точка” для объекта типа Точка, “Вектор” для объекта типа Вектор и ”Прямая” для объекта типа Прямая.

Например,

In[35]:=
```
kmPoint[km, id_String, ___]["id"] := If[id == "", "Точка", id]
```

In[36]:=
```
{P1, P1@"id"}
```

Out[36]=
```
{kmPoint[km, P1Name, {0, 0}], P1Name}
```

3. Функция-свойство с идентификатором “**length**” определена только для геометрического объекта типа Вектор и возвращает его длину

In[37]:=
```
kmVector[km, _, coord_]["length"] := √coord.coord
```

In[38]:=
```
{V1, V1@"length"}
```

Out[38]=
```
{kmVector[km, V1, {1, 1}], √2}
```

4. Функция-свойство с идентификатором “**ort**” определена только для геометрического объекта типа Вектор и возвращает соответствующий ему орт (вектор единичной длины, сонаправленный с исходным вектором)

```
In[39]:= kmVector[km, _, coord_]["ort"] := kmVector["ort", coord / Sqrt[coord.coord]]
```

```
In[40]:= {V1, V2 = V1@"ort", V2@"length"}
```

Out[40]= $\{\text{kmVector}[\text{km}, \text{V1}, \{1, 1\}], \text{kmVector}[\text{km}, \text{ort}, \{\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\}], 1\}$

5. Для геометрического объекта типа Прямая определены функции-свойства с идентификаторами **“coeff”**, **“equ”**, **“normal”**, **“dir”.**

Например,

```
In[41]:= kmLine[km, _, coeffs_]["coeff"] := coeffs;
```

```
In[42]:= kmLine[km, _, coeffs_]["equ"] := coeffs.{x, y, 1} == 0
```

```
In[43]:= {L1, L1["equ"]}
```

```
Out[43]= {kmLine[km, Line1, {-1, 1, 5}], 5 - x + y == 0}
```

Графические образы геометрических объектов

Графический образ геометрического объекта системы “Аналитическая геометрия на плоскости”-- это способ его отображения с помощью графических примитивов.

В Лабораторной работе “Прямая на плоскости” предыдущего семестра был определен оператор **kmGraphics2D,** который преобразовывает геометрические объекты типа Точка, Вектор и Прямая в графические примитивы.

Принадлежность множества точек плоскости одной прямой (коллинеарность точек)

Предположим, что задана последовательность точек в виде списка геометрических объектов типа Точка **points:{__kmPoint}**. Необходимо написать функцию-предикат, которая тестирует принадлежность последовательности Точек **points** некоторой прямой.

In[44]:=
```
kmPointCollinearQ[points : {__kmPoint}] := "Should be specified"
```

Одним из способов задания общего уравнения прямой является использование информации о двух точках P1(x1, y1) и P2(x2, y2), принадлежащих прямой. В этом случае общее уравнение прямой задается в неявной форме соотношением

$$\left|\begin{pmatrix} P-\mathrm{P1} \\ \mathrm{P2}-\mathrm{P1} \end{pmatrix}\right| = 0 \text{ или покоординатно } \left|\begin{pmatrix} x-\mathrm{x1} & y-\mathrm{y1} \\ \mathrm{x2}-\mathrm{x1} & \mathrm{y2}-\mathrm{y1} \end{pmatrix}\right| = 0. \qquad (1)$$

Следствием определения (1) является свойство принадлежности прямой для произвольных трех точек P1, P2 и P3

$$\mathrm{MatrixRank}\left[\begin{pmatrix} \mathrm{P3}-\mathrm{P1} \\ \mathrm{P2}-\mathrm{P1} \end{pmatrix}\right] = 1$$

Последнее равенство можно обобщить для случая произвольного числа точек и сформулировать тест для проверки принадлежности *n* точек одной можно

$$\mathrm{MatrixRank}\left[\begin{pmatrix} \mathrm{P2}-\mathrm{P1} \\ \mathrm{P3}-\mathrm{P1} \\ \ldots \\ \mathrm{Pn}-\mathrm{P1} \end{pmatrix}\right] = 1 \qquad (2)$$

Тест1

Задана последовательность геометрических объектов типа Точка в виде списка

In[45]:=
```
points = {kmPoint["P1", {0, 0}], kmPoint["P2", {0, 1}], kmPoint["P3", {0, 2}]}
```

Out[45]=
```
{kmPoint[km, P1, {0, 0}], kmPoint[km, P2, {0, 1}], kmPoint[km, P3, {0, 2}]}
```

Построим матрицу вида $\begin{pmatrix} \mathrm{P2} \\ \mathrm{P3} \end{pmatrix}$

In[46]:=
```
matrix = #@"coord" & /@ Rest@points
```
Out[46]=
```
{{0, 1}, {0, 2}}
```

Построим матрицу вида $\begin{pmatrix} P2 - P1 \\ P3 - P1 \end{pmatrix}$

In[47]:=
```
matrix = # - (First@points)@"coord" & /@ matrix
```
Out[47]=
```
{{0, 1}, {0, 2}}
```

Проверим принадлежность точек одной прямой

In[48]:=
```
MatrixRank[matrix] == 1
```
Out[48]=
```
True
```

Тест2

Задана последовательность геометрических объектов типа Точка в виде списка

In[49]:=
```
points = {kmPoint["P1", {0, 0}], kmPoint["P2", {0, 1}], kmPoint["P3", {0, 2}], kmPoint["P4", {1, 1}]}
```
Out[49]=
```
{kmPoint[km, P1, {0, 0}], kmPoint[km, P2, {0, 1}], kmPoint[km, P3, {0, 2}], kmPoint[km, P4, {1, 1}]}
```

Построим матрицу вида $\begin{pmatrix} P2 \\ P3 \\ P4 \end{pmatrix}$

In[50]:=
```
matrix = #@"coord" & /@ Rest@points
```
Out[50]=
```
{{0, 1}, {0, 2}, {1, 1}}
```

Построим матрицу вида $\begin{pmatrix} P2 - P1 \\ P3 - P1 \\ P4 - P1 \end{pmatrix}$

In[51]:=
```
matrix = # - (First@points)@"coord" & /@ matrix
```
Out[51]=
```
{{0, 1}, {0, 2}, {1, 1}}
```

Проверим принадлежность точек одной прямой

```
In[52]:= MatrixRank[matrix] == 1
Out[52]= False
```

Результирующая функция

```
In[53]:= kmPointCollinearQ[points : {__kmPoint}] := MatrixRank[# - (Last@points)@"coord" & /@ (#@"coord" & /@ Most@points )] == 1

In[54]:= points = {kmPoint["P1", {0, 0}], kmPoint["P2", {0, 1}], kmPoint["P3", {0, 2}]}
Out[54]= {kmPoint[km, P1, {0, 0}], kmPoint[km, P2, {0, 1}], kmPoint[km, P3, {0, 2}]}

In[55]:= kmPointCollinearQ[#] & /@ {points, Append[points, kmPoint["P4", {1, 1}]]}
Out[55]= {True, False}
```

Взаимное расположение двух прямых на плоскости

Предположим, что заданы две объекта типа Прямая на плоскости **line1_kmLine и line2_kmLine**. Необходимо написать функции-предикаты, которые тестируют взаимное расположение прямых (совпадение, параллельность, ортогональность).

Функция-предикат **kmLineSameQ[line1_kmLine, line2_kmLine]** тестирует совпадение прямых

```
In[56]:= kmLineSameQ[line1_kmLine, line2_kmLine] := "Should be specified"
```

Функция-предикат **kmLineParallelQ[line1_kmLine, line2_kmLine]** тестирует параллельность прямых

```
In[57]:= kmLineParallelQ[line1_kmLine, line2_kmLine] := "Should be specified"
```

Функция-предикат **kmLineOrthogQ[line1_kmLine, line2_kmLine]** тестирует ортогональность прямых

```
In[58]:= kmLineOrthogQ[line1_kmLine, line2_kmLine] := "Should be specified"
```

Совпадение двух прямых на плоскости

Для реализации функции-предиката воспользуемся следующим утверждением

Утверждение 1. Две прямые совпадают, если векторное произведение векторов, состоящих из коэффициентов общего уравнения прямой, равно нулю.

[1] Е. А. Никулин. Компьютерная геометрия и алгоритмы машинной графики. -- СПб.: БХВ-Петербург, 2003.

Векторное произведение векторов вычисляется с помощью встроенной функции **Cross**

```
In[59]:= Cross[line1@"coeff", line2@"coeff"] == {0, 0, 0};

In[60]:= kmLineSameQ[line1_kmLine, line2_kmLine] :=
          Cross[line1@"coeff", line2@"coeff"] == {0, 0, 0}
```

Тестируем

```
In[61]:= line1 = kmLine["line1", x + 2 * y + 3 == 0, {x, y}];
         line2 = kmLine["line2", 2 * x + 4 * y + 6 == 0, {x, y}];

In[63]:= kmLineSameQ[line1, line2]

Out[63]= True
```

Параллельность двух прямых на плоскости

Для реализации функции-предиката воспользуемся следующим утверждением

Утверждение 2. Две прямые параллельны, если их направляющие векторы параллельны (нормальные векторы параллельны).

[1] Е. А. Никулин. Компьютерная геометрия и алгоритмы машинной графики. -- СПб.: БХВ-Петербург, 2003.

Для реализации воспользуемся функцией **isVectorsCollinear** из предыдущего семестра

In[64]:=
```
kmLineParallelQ[line1_kmLine, line2_kmLine] := Or[
  isVectorsCollinear[line1["dir"], line2["dir"]] == 1, (* коллинеарны и сонаправлены *)
  isVectorsCollinear[line1["dir"], line2["dir"]] == -1] (* коллинеарны и противонаправлены *)
```

Ортогональность двух прямых на плоскости

Для реализации функции-предиката воспользуемся следующим утверждением

Утверждение 3. Две прямые ортогональны, если их направляющие векторы ортогональны (нормальные векторы ортогональны).

[1] Е. А. Никулин. Компьютерная геометрия и алгоритмы машинной графики. -- СПб.: БХВ-Петербург, 2003.

Для реализации воспользуемся функцией **isVectorsOrthog** из предыдущего семестра

In[65]:=
```
kmLineOrthogQ[line1_kmLine, line2_kmLine] :=
 isVectorsOrthog[line1["dir"], line2["dir"]]
```

In[66]:=
```
? kmLine*Q
```

▼ Global`

kmLineOrthogQ	kmLineParallelQ	kmLineSameQ

Взаимное расположение точки и прямой на плоскости

Предположим, что задан объект типа Точка **point_kmPoint** и объект типа Прямая **line_kmLine**. Необходимо написать функцию, которая возвращает расстояние от точки до прямой с точностью до знака, где знак определяет, в какой из полуплоскостей относительно заданной прямой **line_kmLine** расположена точка **point_kmPoint**.

```
In[67]:= kmDistance[point_kmPoint, line_kmLine] := "Should be specified"
```

Расстояние от точки P(x1, y1) до прямой, заданной общим уравнением $A\,x + B\,y + C == 0$, вычисляется по формуле

$$d = \frac{|\,A\,x1 + B\,y1 + C\,|}{|\,N\,|}, \tag{3}$$

где в знаменателе стоит длина нормального вектора к прямой.

[1] Е. А. Никулин. Компьютерная геометрия и алгоритмы машинной графики. -- СПб.: БХВ-Петербург, 2003.

Реализация

Добавим новую функцию-свойство для геометрических объектов типа Линия с идентификатором “**value**”, которая возвращает значение вида $A\,x + B\,y + C$ для произвольной точки $P(x, y)$, заданной списком координат

```
In[68]:= kmLine[km, _, coeff_List]["value", point_List] :=
          coeff.Append[point, 1]
```

Протестируем работу введенной функции

```
In[69]:= line = kmLine["line", x - y == 0, {x, y}];
         points = {kmPoint["P1", {0, 0}], kmPoint["P2", {0, 1}], kmPoint["P3", {1, 0}]};

In[71]:= line["value", #@"coord"] & /@ points

Out[71]= {0, -1, 1}
```

Результирующая функция вычисления расстояния от точки до прямой с точностью до знака

```
In[72]:= kmDistance[point_kmPoint, line_kmLine] :=
           line["value", point@"coord"]
           ----------------------------
             line["normal"]@"length"
```

Пересечение двух прямых

Важнейшей задачей геометрии на плоскости является расчет точки пересечения двух прямых.

[1] Е. А. Никулин. Компьютерная геометрия и алгоритмы машинной графики. -- СПб.: БХВ-Петербург, 2003.

Предположим, что заданы две прямые на плоскости **line1_kmLine и line2_kmLine**. Необходимо написать функцию-конструктор для геометрического объекта типа Точка, которая создает точку пересечением двух заданных прямых.

Учтем при построении конструктора, что точка пересечения существует и единственна, если прямые не совпадают и не параллельны.

In[73]:=
```
kmPoint[line1_kmLine, line2_kmLine] :=
 "Should be specified" /;
  And[! kmLineSameQ[line1, line2], ! kmLineParallelQ[line1, line2]]
```

Создадим два объекта типа Линия

In[74]:=
```
line1 = kmLine["line1", x - y == 0, {x, y}]; line2 = kmLine["line2", x + y == 0, {x, y}];
```

Для нахождения точки пересечения двух прямых составим систему линейных алгебраических уравнений из общих уравнений прямых и решим систему

In[75]:=
```
{x, y} /. First@Solve[{line1@"equ", line2@"equ"}, {x, y}]
```

Out[75]=
```
{0, 0}
```

Результирующая функция-конструктор

In[76]:=
```
kmPoint[line1_kmLine, line2_kmLine] :=
 kmPoint["point", {x, y} /. First@Solve[{line1@"equ", line2@"equ"}, {x, y}]] /;
  And[! kmLineSameQ[line1, line2], ! kmLineParallelQ[line1, line2]]
```

Так как функция kmLineParallelQ на лекции не определена, то новый конструктор протестировать нельзя.

Важные понятия

В рамках темы “Взаимное расположение геометрических объектов типа Точка, Вектор, Прямая” функционал системы “Аналитическая геометрия на плоскости” расширен следующими функциями

1. Функция-предикат **kmPointCollinearQ** тестирует принадлежность списка точек некоторой прямой

In[77]:=
```
? kmPointCollinearQ
```

```
Global`kmPointCollinearQ

kmPointCollinearQ[points : {__kmPoint}] := MatrixRank[(#1 - Last[points][coord] &) /@ (#1[coord] &) /@ Most[points]] == 1
```

2. Функции-предикаты **kmLineOrthogQ, kmLineParallelQ, kmLineSameQ** тестируют взаимное расположение прямых (совпадение, параллельность, ортогональность)

In[78]:=
```
? kmLine*Q
```

▼ Global`

kmLineOrthogQ	kmLineParallelQ	kmLineSameQ

3. Функция-свойство для геометрических объектов типа Линия с идентификатором “**value**”, которая возвращает значение вида $A\,x + B\,y\,+C$ для произвольной точки $P(x,\,y)$, заданной списком координат

In[79]:=
```
? kmLine
```

```
Global`kmLine
```

```
kmLine[km, _, coeffs_][coeff] := coeffs

kmLine[km, _, coeffs_][equ] := coeffs.{x, y, 1} == 0

kmLine[km, _, coeff_List][value, point_List] := coeff.Append[point, 1]

kmLine[id_String, C_. + A_. x_ + B_. y_ == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {A, B, C}]

kmLine[id_String, C_. + B_. y_ == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {0, B, C}]

kmLine[id_String, C_. + A_. x_ == 0, {x_Symbol, y_Symbol}] := kmLine[km, id, {A, 0, C}]

kmLine[id_String, P0_kmPoint, Dir_kmVector] := kmLine[id, Det[{Dir[coord], {x, y} - P0[coord]}] == 0, {x, y}]

kmLine[id_String, P0_kmPoint, P1_kmPoint] := kmLine[id, P0, kmVector[dir, P1[coord] - P0[coord]]]

kmLine[id_String, P0_kmPoint, {normal, norm_kmVector}] := Should be specified
```

4. Функция **kmDistance** возвращает расстояние от точки до прямой с точностью до знака, где знак определяет, в какой из полуплоскостей относительно заданной прямой расположена точка

In[80]:= **? kmDistance**

```
Global`kmDistance
```

```
kmDistance[point_kmPoint, line_kmLine] := line[value,point[coord]] / line[normal][length]
```

5. Функция-конструктор для геометрического объекта типа Точка, которая создает точку пересечением двух заданных прямых

In[81]:= **? kmPoint**

```
Global`kmPoint

kmPoint[_, _, coords_][coord] := coords

kmPoint[km, id_String, ___][id] := If[id == , Точка, id]

kmPoint[id_String, {x_, y_}] := kmPoint[km, id, {x, y}]

kmPoint[id_String, {ρ_, ϕ_}, pol] := kmPoint[km, id, {ρ Cos[ϕ], ρ Sin[ϕ]}]

kmPoint[line1_kmLine, line2_kmLine] :=
 kmPoint[point, {x, y} /. First[Solve[{line1[equ], line2[equ]}, {x, y}]]] /; ! kmLineSameQ[line1, line2] && ! kmLineParallelQ[line1, line2]
```