КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ
И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра информационных систем

Ч.Б.МИННЕГАЛИЕВА

КОМПЬЮТЕРНАЯ ГЕОМЕТРИЯ И ГРАФИКА В СИСТЕМАХ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ

Казань – 2021

УДК 004.92, 004.94

Принято на заседании учебно-методической комиссии ИВМиИТ
Протокол от 18 марта 2021 года

Рецензенты:
кандидат физико-математических наук,
доцент кафедры информационных систем КФУ **Л.Э.Хайруллина**
старший преподаватель кафедры информационных систем КФУ
М.М.Аюпов

Миннегалиева Ч.Б.
Компьютерная геометрия и графика в системах компьютерной математики / Ч.Б.Миннегалиева. – Казань: Казан. ун-т, 2021. – 30 с.

Учебно-методическое пособие посвящено изучению возможностей систем компьютерной математики для решения задач геометрии и графики. Оно адресовано студентам, обучающимся по направлению «Информационные системы и технологии», а также читателям, интересующимся проблемами компьютерной геометрии и графики, работой в системах компьютерной математики

ОГЛАВЛЕНИЕ

1. ГРАФИКА В СИСТЕМЕ MATHEMATICA

С середины 60-х годов XX века началась эра систем компьютерной математики (СКМ), CAS – Computer algebra system. Mathematica – система компьютерной алгебры, используемая при решении многих научных, инженерных, математических задач. Была придумана Стивеном Вольфрамом, в настоящее время разрабатывается компанией Wolfram Research. В течение трёх десятилетий компания постепенно создала базу технологий, которая делает возможным обширный портфель инновационной продукции. В центре находится язык Wolfram Language, который определяет уникальное сближение компьютерных вычислений и информации. Компания Wolfram предоставляет ресурсы, посвященные вычислениям и знаниям, которые находятся в открытом пользовании. Один из них – Wolfram Language Sandbox, где можно бесплатно писать код.

1.1. Визуализация функций

В Mathematica существуют встроенные функции для построения графиков: ArrayPlot, ContourPlot, DateListLogPlot, DateListPlot, DensityPlot, DiscretePlot, GraphPlot, LayeredGraphPlot, ListContourPlot и другие.

Все указанные функции имеют определённую структуру. Они имеют два обязательных аргумента и один необязательный. Первый обязательный аргумент есть выражение Mathematica, которое определяет зависимость (или зависимости), которую (которые) требуется построить. Второй аргумент определяет аргументы функций и пределы их изменения. Необязательные аргументы графических функций – опции.

Графики в Mathematica являются графическими объектами, которые создаются (возвращаются) соответствующими графическими функциями. Они охватывают построение практически всех типов графиков. Достигается это за счет применения опций и директив графики.

Mathematica допускает такие конструкции, как:

• Plot [Sin [x],{x, 0, 20}] – построение синусоиды;

• g: = Plot [Sin [x],{x, 0, 20}] – задание объекта –синусоиды

с отложенным выводом;

• g = Plot [Sin [x],{x, 0, 20}] – задание объекта – синусоиды

с немедленным выводом.

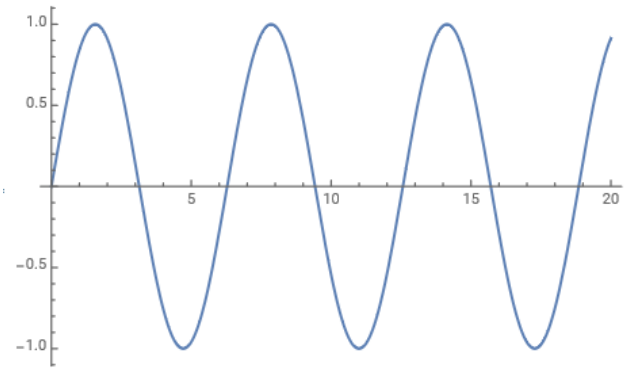


Рис. 1.1. График функции $y = \sin x$

Опции определяют стиль оформления, дополнительные параметры и элементы рисунков для повышения их наглядности и информативности. Опции, например, функции Plot можно узнать с помощью команды Options [Plot]. На рисунке 1.2. приведен график функции, к которому применены две опции: толщина линии и диапазон построения графика.

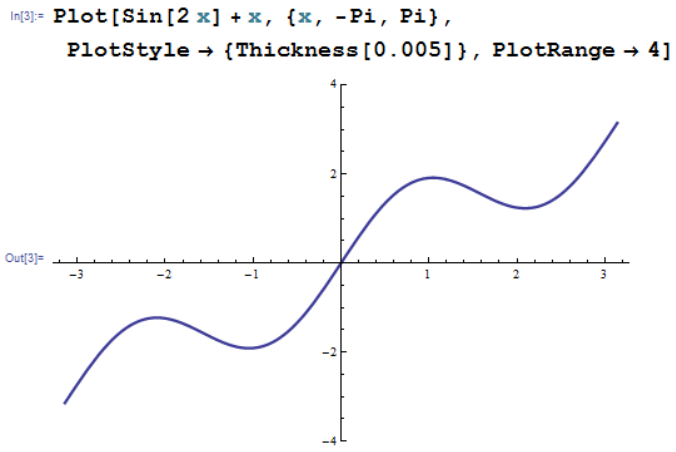


Рис. 1.2. График функции, построенный с использованием опций

Язык Wolfram Language включает в себя множество оригинальных алгоритмов, которые автоматизируют создание 2D- и 3D-визуализаций.

Для построения поверхностей используются функции Plot3D, ContourPlot3D, ListPlot3D, ListDensityPlot3D и другие. Указываются функция и пределы изменения переменных:

Plot3D[f, {x, x_{min} , x_{max}}, {y, y_{min} , y_{max}}]

Встроенные функции Mathematica для работы с трёхмерной графикой оканчиваются на Plot3D. Это такие функции, как Plot3D, ListPlot3D, ParametricPlot3D, VectorPlot3D и другие.

Среди опций функции Plot3D есть Mesh – координатная сетка на поверхности, Background – определяет задний фон, MaxRecursion – определяет участки поверхности, где изменение значений больше, чем на других участках и другие.

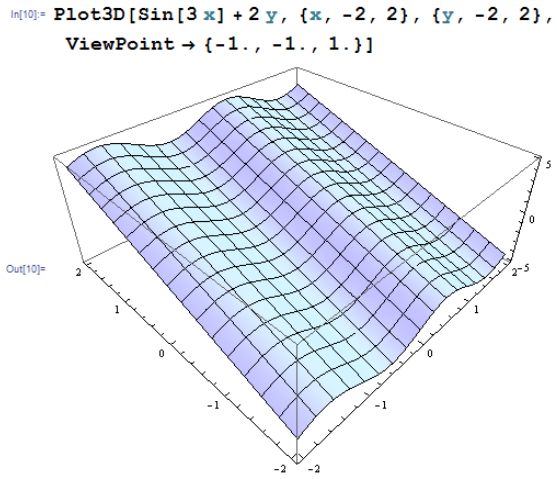


Рис. 1.3. Пример использования Plot3D

1.2. Примитивы графики

Примитивами двумерной графики называют дополнительные указания, вводимые в функцию Graphics для построения некоторых заданных геометрических фигур. Функция Graphics [primitives, options] – представляет двумерное графическое изображение. Применение примитивов в составе функции Graphics избавляет пользователя от задания математических выражений, описывающих эти фигуры.

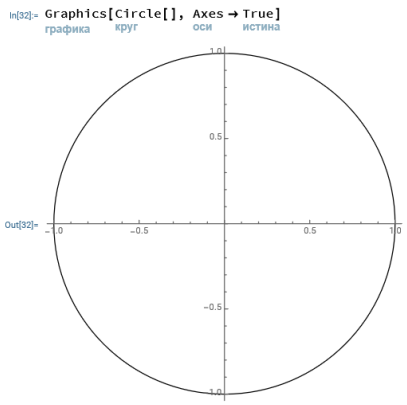


Рис. 1.4. Пример использования Graphics

Примеры примитивов двумерной графики:

Circle [{x, y}, r] – строит окружность с радиусом r и центром в точке {x, y}.

Circle [{x, y}, {rx, ry}] – строит эллипс с центром {x,y}и полуосями rx и ry.

Disk [{x, y}, {rx, ry}] – строит закрашенный овал с полуосями rx и ry и центром {x,y}.

Disk [{x, y}, r, {theta1, theta2}] – строит сегмент круга радиуса r с центром {x,y} и углами концевых точек theta1 и theta2.

Line [{pt1, pt2, ...}] – строит линию, соединяющую последовательность точек.

Point [{x, y}] – строит точку с координатами x и y.

Polygon [{x1, y1},{x2, y2},...] – построение полигона с закраской.

Объединить построенные разным способом графики возможно при помощи Show.

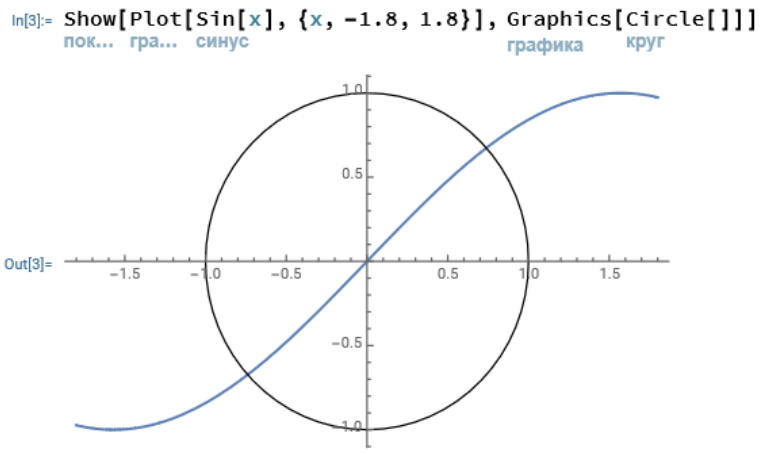


Рис. 1.5. Пример использования Show

1.3. Элементы управления

В Mathematica имеются многочисленные элементы управления: Slider, Checkbox, RadioButton и т. д. Эта особенность делает возможным организацию удобного интерфейса для пользователя. Функция Manipulate [ ] позволяет визуализировать зависимость выражения от параметров.

Пример. Строится график функции, аргумент x меняется в пределах от -5 до 5. А n является параметром, принимает значения от 0,1 до 5. В зависимости от значения n строится график функций sin x, sin 1,5 x и т.д.

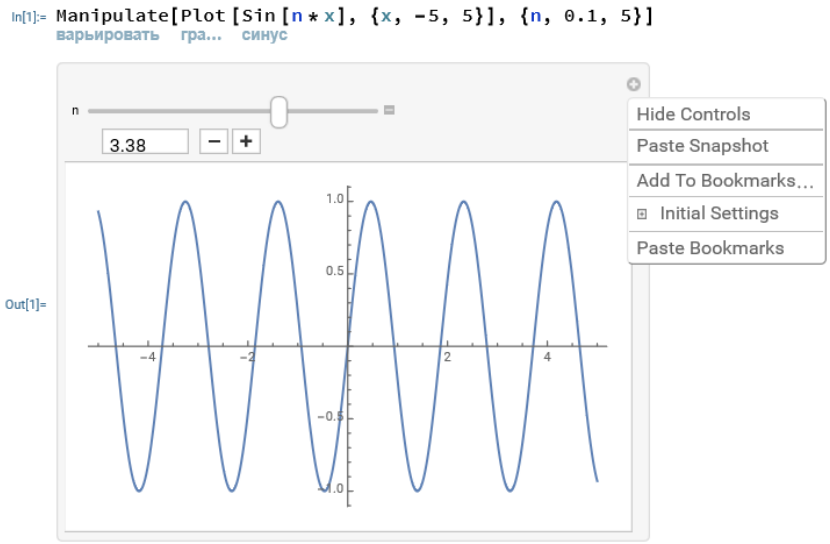


Рис. 1.6. Пример использования Manipulate

Mathematica по умолчанию выбирает элемент управления или пользователь может написать конкретный вид. В примере указывается, что для параметра m необходимо использовать элемент управления RadioButtonBar, для параметра k – SetterBar. Параметр n меняется с помощью Slider (рисунок 1.7).

```
In[2]:= Manipulate[Plot[m * Sin[n * x] + k, {x, 0, 5},
        варьировать  график ...  синус

    PlotRange → {-4, 8}, PlotStyle → {Black, Thickness[0.005]}],
    отображаемый диапазон...  стиль графика  чёрный  толщина

  {n, 0.1, 5}, {k, {1, 1.5, 2, 2.5}, ControlType → SetterBar},
                                  тип элемента уп...  ряд установочных элементов

  {m, {1, 2, 3}, ControlType → RadioButtonBar}]
                  тип элемента уп...  ряд радиокнопок
```

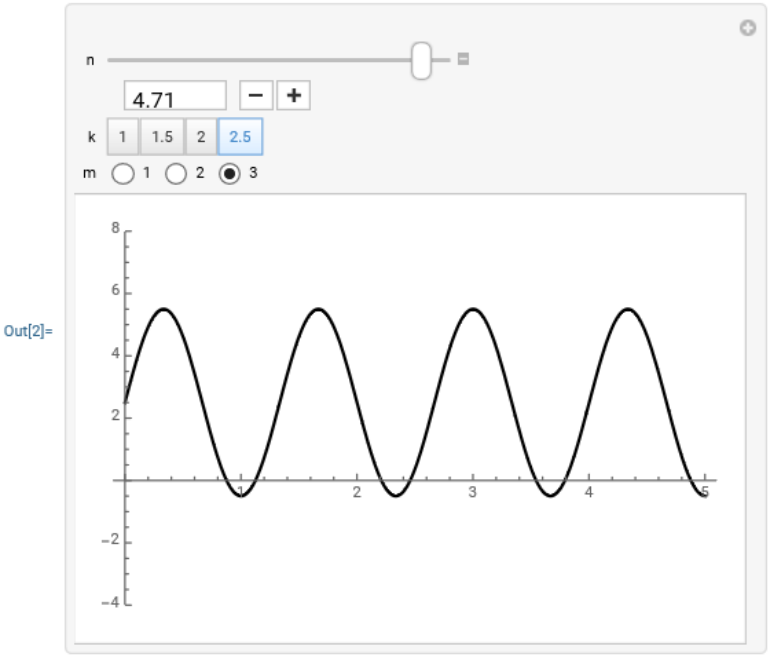


Рис. 1.7. Элементы управления Manipulate

Manipulate [expr, {{u, uinit}, Locator}] – параметр u задается с помощью локатора Locator, начальное положение локатора – uinit. Локатор позволяет интерактивно изменять позицию элемента на графике. В данном случае мы сможем изменить положение точки, «захватив» ее мышкой. В примере на рисунке 1.7. абсцисса и ордината точки могут меняться в пределах от -3 до 3. (1, 2) – начальное положение точки.

```
In[10]:= Manipulate[Graphics[{Line[{{0, 0}, l[[1]]}]},
           Axes → True, PlotRange → 3],
         {{l, {{1, 2}}}, {-3, -3}, {3, 3}, Locator}]
```

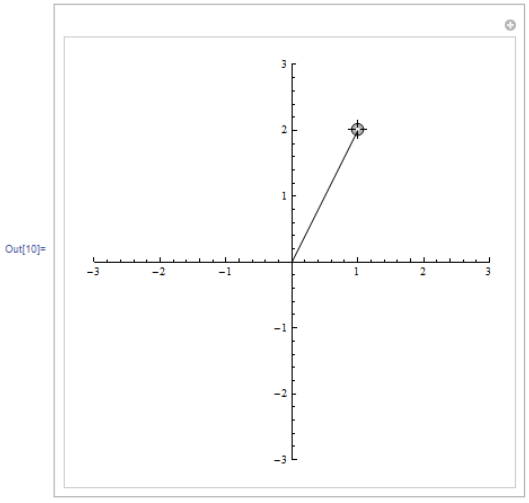


Рис. 1.7. Пример использования Locator

В коде ниже указаны три линии, соединяющие вершины треугольника (Line). Одна вершина – точка (0, 0), две другие – v_1 и v_2 определяются с помощью Locator. Задана функция S, вычисляющая площадь треугольника. В этом примере угол между сторонами треугольника находится через скалярное произведение векторов, имеющих координаты, определяемые с помощью Locator (результат приведен на рисунке 1.8).

```
In[19]:= Manipulate[
  Graphics[{{Line[{{0, 0}, v1}]}, {Line[{v1, v2}]},
    {Line[{v2, {0, 0}}]}}, PlotRange → {{-5, 5}, {-5, 5}},
   ImageSize → 400, Axes → True,
   PlotLabel → "Площадь треугольника равна "
     NumberForm[S[v1, v2], {5, 2}], TextStyle → {Black, 16}],
  {{v1, {1, 2}}, {-5, -5}, {5, 5}, Locator},
  {{v2, {-4, 1}}, {-5, -5}, {5, 5}, Locator},
  Initialization :↦
   (S[a_, b_] :=
     0.5 * Sqrt[a.a] * Sqrt[b.b] *
       Sin[ArcCos[(a.b) / (Sqrt[a.a] * Sqrt[b.b])]] // N)]
```

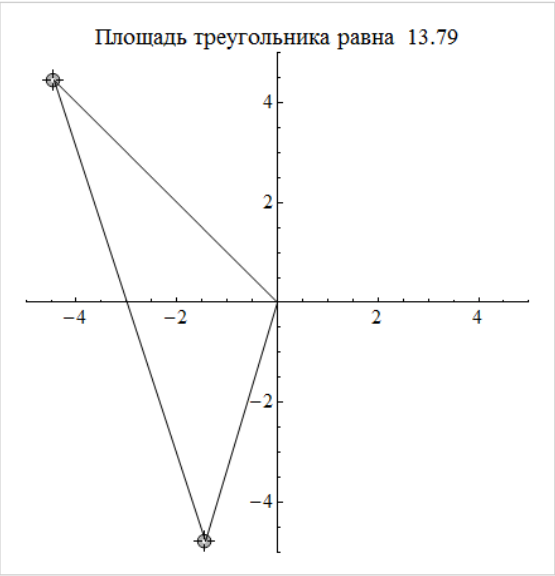


Рис. 1.8. Нахождение площади треугольника

2. КРИВЫЕ БЕЗЬЕ. ГРАФЫ

2.1. Работа с кривыми Безье (Bezier Curve)

Метод де Кастельжо (Кастелье, de Casteljau) основан на разбиении отрезков, соединяющих исходные точки в отношении t (значение параметра), а затем в рекурсивном повторении этого процесса для полученных отрезков.

Обозначим опорные точки как $P_i \quad i = 0,1,...,m$. Начало кривой в точке P_0 (t=0), конец в точке P_m (t=1), для каждого t найдем точку P(t).

Случай m=1: $P(t) = (1-t)\cdot P_0 + t\cdot P_1$

Случай m=2:

$$P_0^1 = (1-t)P_0 + tP_1 \quad P_1^1 = (1-t)P_1 + tP_2$$

и т.д.

Кривая Безье с 4 опорными точками.

$$P_0^1 = (1-t)P_0 + tP_1 \quad P_1^1 = (1-t)P_1 + tP_2 \quad P_2^1 = (1-t)P_2 + tP_3$$

Рекуррентная формула де Кастелье для вычисления кривой Безье:

$$r_i(t,k) = (1-t)r_i(t;k-1) + tr_{i+1}(t;k-1)$$

Начиная от значений $r_i(t;0) = p_i$, получаем последовательно значения $r_i(t;k) \quad 1 \le k \le n.$

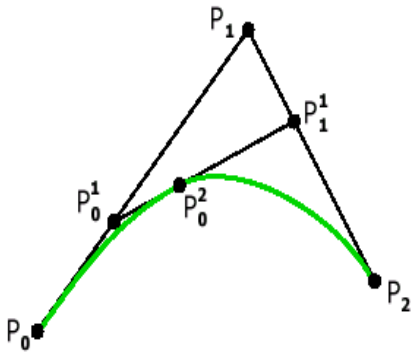


Рис. 2.1. Построение кривой Безье

В системе Mathematica есть специальная функция BezierCurve для работы с кривыми Безье:

```
In[13]:= pts = {{0, 0}, {1, 1}, {2, -1}, {3, 0}};
         Graphics[{BezierCurve[pts], Green, Line[pts], Red, Point[pts]}]
```

Out[14]=

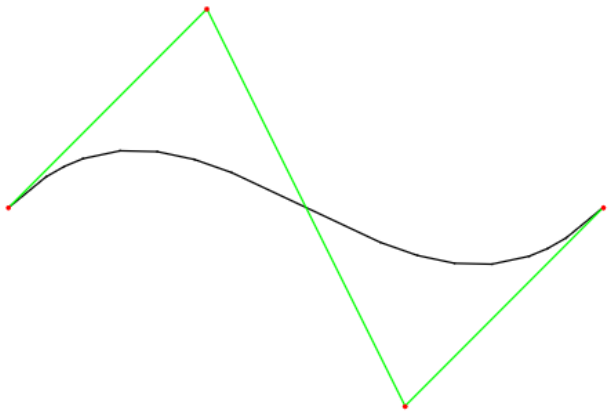


Рис. 2.2. Пример использования BezierCurve

Ниже приведен пример работы студентов, интерактивная демонстрация построения кривых Безье.

```
p = {{0.0, 0.5}, {0.5, 2.5}, {3.0, 2.0}, {4.0, 0.0}};
b[t_, i_, r_] := (1 - t) * b[t, i, r - 1] + t * b[t, i + 1, r - 1];
b[t_, 0, 1] := (1 - t) * p[[1, All]] + t * p[[2, All]]
b[t_, 1, 1] := (1 - t) * p[[2, All]] + t * p[[3, All]]
b[t_, 2, 1] := (1 - t) * p[[3, All]] + t * p[[4, All]]
Manipulate[Module[{plt}, plt = ParametricPlot[b[t, 0, 3], {t, 0, s}, PlotStyle → {Red,
      Thickness[0.004]}][[1]];
  p2 = {b[s, 0, 1], b[s, 1, 1], b[s, 2, 1]};
  p3 = {b[s, 0, 2], b[s, 1, 2]};
  Graphics[{plt, {Gray, Line[p]}, {PointSize[0.02], Point[p]},
    {Green, Line[p2]},
    {Darker[Green], PointSize[0.02], Point[p2]},
    {Blue, Line[p3]},
    {Darker[Blue], PointSize[0.02], Point[p3]}, {Darker[Red], PointSize[0.02],
      Point[b[s, 0, 3]]}}, PlotRange → {{-0.1, 4.1}, {-0.1, 2.6}}, ImageSize → {500,
     375}]], {{s, 0.5, "t"}, 0.01, 1}]
```

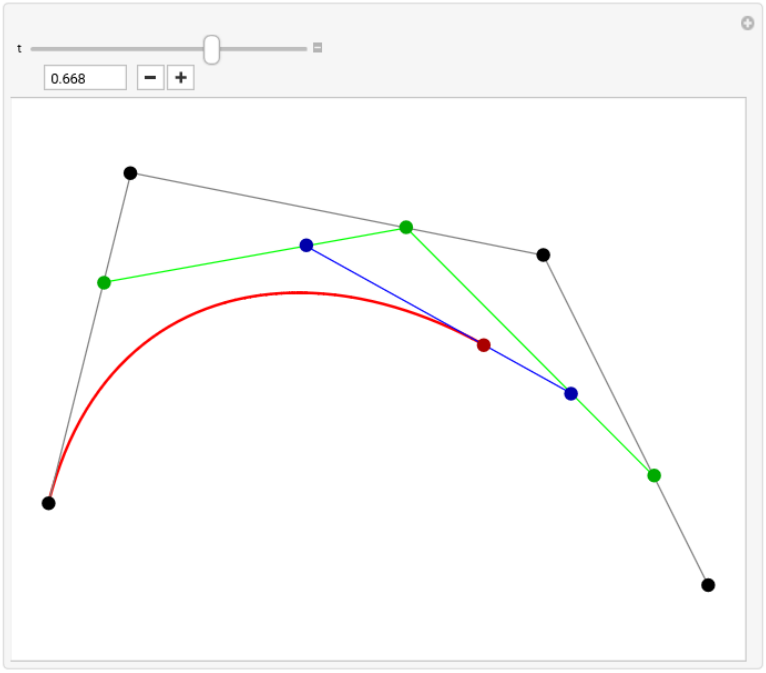


Рис. 2.3. Построение кривой Безье

2.2. Работа с графами

Объекты, допускающие интерпретацию на языке теории графов, встречаются в математике и ее приложениях. Компьютерная геометрия – не исключение. При этом в ней графы выступают и как техническое средство (например, сетка линий на поверхности), и как объект изучения (прежде всего, оптимизационные задачи на графах).

Графом называется пара G = (V, E), где V – некоторое конечное множество, элементы которого называются вершинами, а E – семейство неупорядоченных пар (двухэлементных подмножеств) множества V, называемых ребрами. Такие графы называют простыми. Аналогично определяются ориентированные графы (ребра – упорядоченные пары вершин) и мультиграфы (допускаются петли и кратные, т. е. соединяющие одни и те же вершины, ребра).

Для изображения графов используется функция GraphPlot. Примеры использования данной функции приведены на рисунках ниже.

```
GraphPlot[{{0, 1, 1}, {1, 0, 1}, {1, 1, 0}}]
```

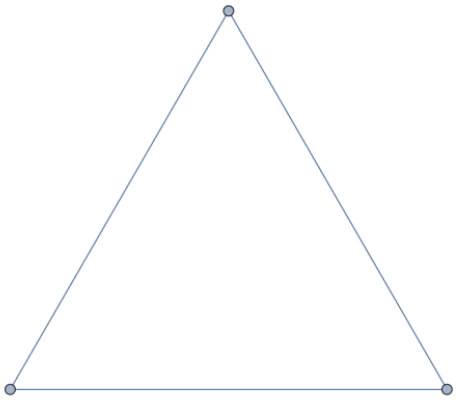


Рис. 2.4. Изображение графа, заданного с помощью матрицы смежности

```
GraphPlot[{1 → 2, 2 → 3, 3 → 4, 4 → 1, 2 → 4},
  DirectedEdges → True]
```

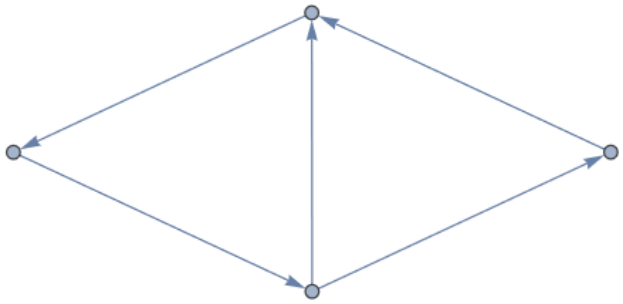


Рис. 2.5. Изображение ориентированного графа

```
GraphPlot[Table[i ⟼ Mod[i + 1, 3], {i, 8}],
  GraphLayout → "CircularEmbedding"]
```

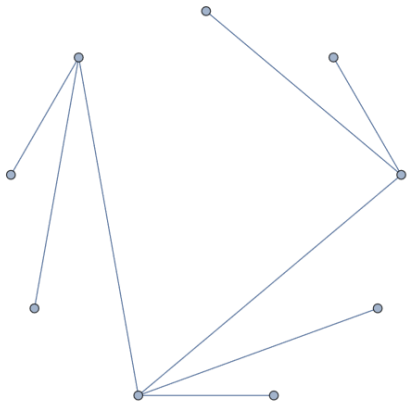


Рис. 2.6. Пример использования GraphPlot

Также используются функции Graph, GraphPlot3D, LayeredGraphPlot и другие.

3.РАБОТА С ИЗОБРАЖЕНИЯМИ

3.1. Обзор функций Wolfram Language

Поддержка работы с изображениями интегрирована с мощными математическими и алгоритмическими возможностями Wolfram Language.

Первая группа функций – это Creating & Importing Images (Создание и импорт изображений). При помощи данных команд можно импортировать изображения, создавать их, захватить изображение или видео в режиме реального времени с камеры или другого устройства.

При помощи функций группы Image Properties (Свойства изображения) можно получить указанные свойства изображения, срезы 2D-изображения для 3D-изображения, позиции указанного значения пикселя.

Функции (команды) группы Basic Image Manipulation предназначены для базовых операций с изображениями. Это такие функции, как обрезка, удаление границ, добавление границ, нахождение границ. Также в эту группы включены геометрические операции – изменение размера, поворот, отражение изображения. Команды Image Composition предоставляют возможность объединения изображений, создания коллажа, наложения изображений. В эту же группу включены такие базовые операции, как настройка уровней, яркости, резкости. Pixel Operations работают с позициями пикселей.

Отразить цвета, смешать их, получить гистограммы, работать со цветовыми пространствами можно при помощи функций группы Color Manipulation.

Команды группы Morphological Image Processing позволяют провести морфологическую обработку изображений. Например, функция MorphologicalComponents дает массив, где каждый пиксель изображения заменяется целочисленным индексом, представляющим область изображения, в котором находится пиксель. MorphologicalPerimeter выделяет периметр областей переднего плана на изображении.

Группа Computer Vision – это функции для идентификации изображений и распознавания объектов, а также извлечения признаков. Язык Wolfram Language поддерживает определенные геометрические особенности, такие как края и углы, а также общие ключевые точки, которые можно использовать для обнаружения и сравнения изображений. Например, FindFaces даёт возможность найти человеческие лица на изображении.

Команды группы Image Restoration – язык Wolfram Language не только включает в себя высокооптимизированные реализации стандартных фильтров восстановления изображений, но также предоставляет сложные функции и алгоритмы, позволяющие ретушировать, снимать шум и размыть изображения, используя самые современные методы.

Необходимо отметить, что деление функций (команд) на группы – условное. Например, команда GaussianFilter может быть использована и при анализе изображений и при их реставрации.

3.2. Примеры работы с командами Images

Примеры с кодом приведены на https://www.wolfram.com/

Пример 1. FacialFeatures – возвращает минимальную сводку черт лица для всех обнаруженных лиц на изображении.

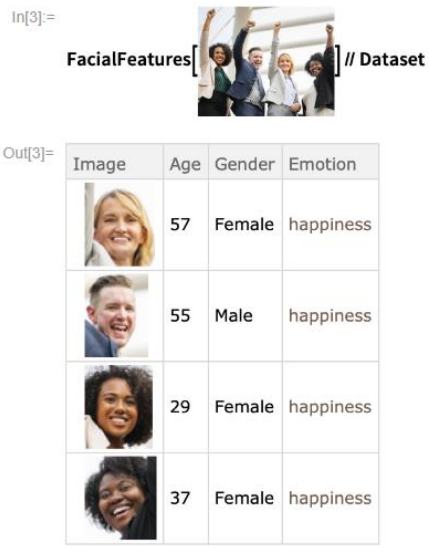


Рис. 2.7. Пример использования FacialFeatures

Пример 2. ImageHistogram строит гистограмму уровней пикселей для каждого канала в изображении. Например, гистограмма RGB изображения:

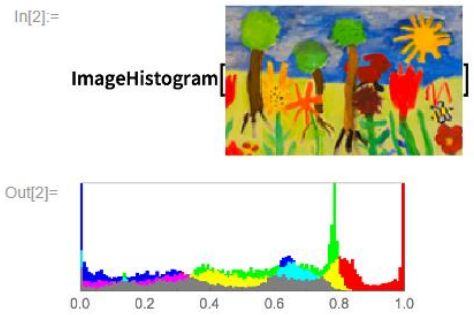


Рис. 2.8. Пример использования ImageHistogram

Или гистограмма изображения в градациях серого:

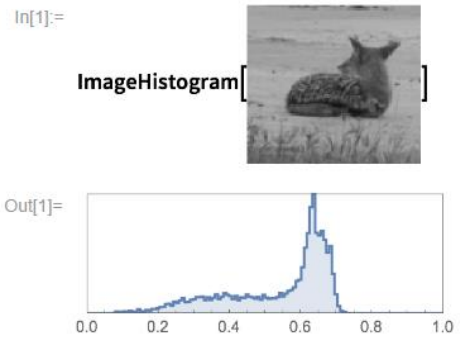


Рис. 2.8. Гистограмма изображения в градациях серого

Пример 3. MorphologicalPerimeter применяется для выявления границ связных областей

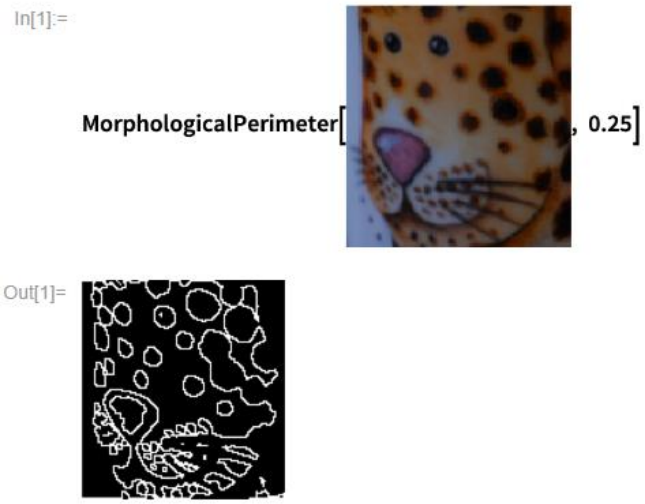


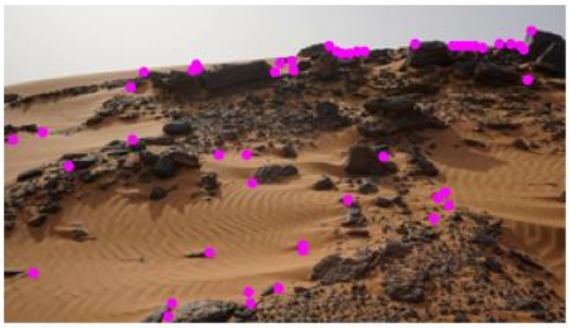
Рис. 2.9. Работа с MorphologicalPerimeter

Пример 4. ImageKeypoints находит ключевые элементы изображения и возвращает их координаты.

```
In[2]:= ImageKeypoints[i, {"Position", "Strength", "Scale"},
          Method → "FAST", MaxFeatures → 50] // Short

{{{264.5, 148.5}, 0.67451, 3.5}, ≪48≫, {{130.5, 16.5}, ≪19≫, 3.5}}

HighlightImage[i, ImageKeypoints[i, Method → "FAST", MaxFeatures → 50]]
```



```
HighlightImage[i, ImageKeypoints[i, Method → "AGAST", MaxFeatures → 50]]
```

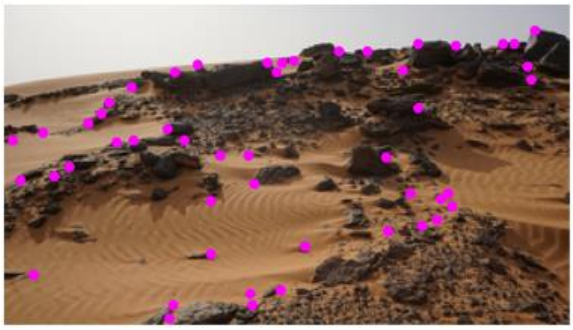


Рис. 2.9. Работа с ImageKeypoints

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

Задания к теме 1.

1) а) построить график функции $f(x)$, использовав любые 5 опций;

б) построить поверхность, которая определяется функцией $g(x, y)$, применить любые 5 опций, объяснить результаты.

номер варианта	$f(x)$	$g(x, y)$
1	$x \cdot \cos 3x$	$\sin(x + y)$
2	$x \cdot \ln \frac{3}{x}$	$\cos(x + y)$
3	$x + \cos 3x$	$tg(x + y)$
4	$x \cdot \cos(x + 1)$	$\sin(x - y)$
5	$x \cdot \cos(3^x)$	$\sin(x^2 - y)$
6	$x \cdot \sin 3x$	$\sin(x - y^2)$
7	$x \cdot \sin(2^x)$	$\sin(x^2 - y^2)$
8	$\ln 3x - x$	$\sin(x^2 + y^2)$
9	$x \cdot \cos \frac{3x - 4}{5}$	$\sin(x + 3y)$
10	$\sin 2^x - x$	$\sin(2x - 3y)$
11	$x \cdot \cos(3 - x)$	$\sin(2x - 3^y)$
12	$\cos x^4$	$\sin(2^x - 3y\ )$

2) Использование функции Manipulate

Показать зависимость графика $f(x)$ от параметров a, b, c, используя различные элементы управления: Slider, RadioButtonBar, SetterBar. Диапазоны

или набор значений параметров a, b, c подобрать наиболее полно демонстрирующими поведение функции.

номер варианта	$f(x)$
1	$ax \cdot \cos bx + c$
2	$ax \cdot \ln \frac{b}{x} - c$
3	$a \cos bx + c$
4	$ax \cdot \cos(bx + 1) + c$
5	$ax \cdot \cos(3^x) + bx + c$
6	$ax \cdot \sin 3x + bx + c$
7	$ax \cdot \sin(2^x) + bx + c$
8	$a \ln bx - x + c$
9	$ax \cdot \cos \frac{bx - 4}{5} + c$
10	$a \sin 2^x - bx + c$
11	$ax \cdot \cos(3 - x) + bx - c$
12	$ax \cos(x^4) + bx - c$

3) Выполнить задание, используя Locator. Объяснить работу

Вариант 1. Построить два вектора с началом в точке (0, 0), так чтобы пользователь мог изменять положение точек, являющихся концами векторов. Выводить текущее значение угла между векторами.

Вариант 2. Построить отрезок так, чтобы пользователь мог изменять положение точек, являющихся концами отрезка. Выводить текущее значение абсциссы середины отрезка.

Вариант 3. Построить вектор с началом в точке (0, 0) так, чтобы пользователь мог изменять положение точек, являющихся концами вектора. Выводить текущие положения точек, являющихся проекциями вектора на оси координат и текущие координаты вектора.

Вариант 4. Построить четырехугольник, две вершины которого находятся на оси абсцисс, положение двух других может менять пользователь. Выводить текущее значение периметра четырехугольника.

Вариант 5. Построить четырехугольник, две вершины которого находятся на оси ординат, положение двух других может менять пользователь. Выводить текущие значения длин диагоналей.

Вариант 6. Построить треугольник, две вершины которого находятся на оси абсцисс, положение точки, являющейся третьей вершиной, может менять пользователь. Провести медиану к стороне, принадлежащей оси абсцисс. Выводить текущую длину медианы.

Вариант 7. Построить два вектора с началом в точке (0, 0) так, чтобы пользователь мог изменять положение точек, являющихся концами векторов. Выводить текущее значение площади параллелограмма, построенного на данных векторах.

Вариант 8. Построить вектор с началом в точке (0, 0) так, чтобы пользователь мог менять положение конца вектора. Выводить текущее значение угла между вектором и осью абсцисс.

Вариант 9. Построить треугольник, две вершины которого находятся на оси абсцисс, положение точки, являющейся третьей вершиной, может менять пользователь. Выводить текущее значение периметра треугольника.

Вариант 10. Построить треугольник, одна вершина которого находится на оси абсцисс, положение двух других может менять пользователь. Выводить текущее значение периметра треугольника.

Вариант 11. Построить два вектора с началом в точке (0, 0), так чтобы пользователь мог изменять положение точек, являющихся концами векторов. Выводить текущее значение длины вектора – суммы данных векторов.

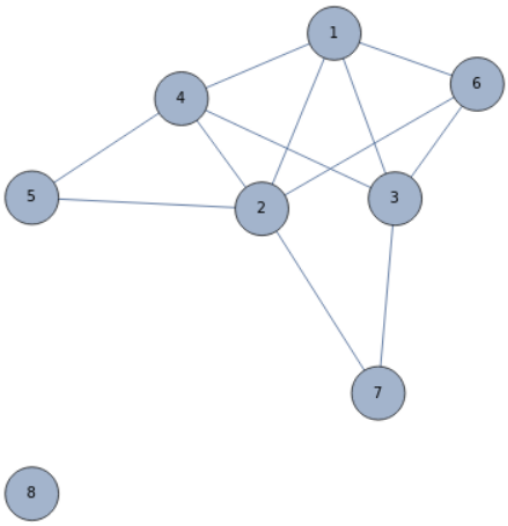
Вариант 12. Построить треугольник, две вершины которого находятся на оси абсцисс, положение точки, являющейся третьей вершиной, может менять пользователь. Выводить текущее значение высоты треугольника, проведенной из этой вершины.

Задания к теме 2

1) Изучить материал по теме 2 (Кривые Безье). Приготовиться объяснить приведенный код. Посмотреть по справке встроенную функцию BezierCurve. Применить ее к данным точкам, сравнить результаты. Показать вариант изменения кода для 5 точек.

2) Создать случайный граф с 8 вершинами и 12 ребрами:

```
g = RandomGraph[{8, 12}, VertexLabels → Placed[Automatic, Center], VertexSize → 0.4]
```



Узнать степени вершин графа:

```
VertexDegree[g]
```

```
{4, 5, 4, 4, 2, 3, 2, 0}
```

Найти кратчайший путь от одной вершины до другой (выбрать любые):

```
FindShortestPath[g, 4, 6]
```

```
{4, 1, 6}
```

Найти расстояние от исходной вершины до целевой вершины (выбрать любые):

```
GraphDistance[g, 4, 6]
```

```
2
```

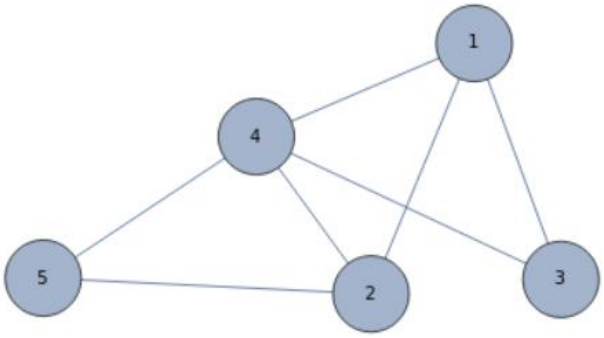
Найти связанные компоненты графа:

```
ConnectedComponents[g]
```

```
{{1, 2, 3, 4, 6, 5, 7}, {8}}
```

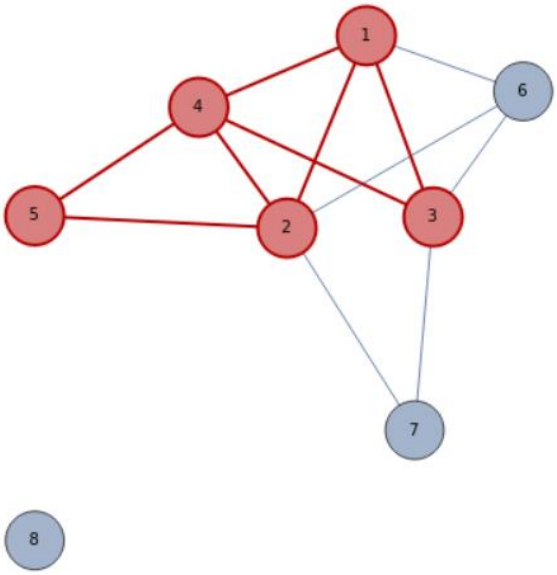
У произвольной вершины найти соседние:

```
g1 = NeighborhoodGraph[g, 4]
```

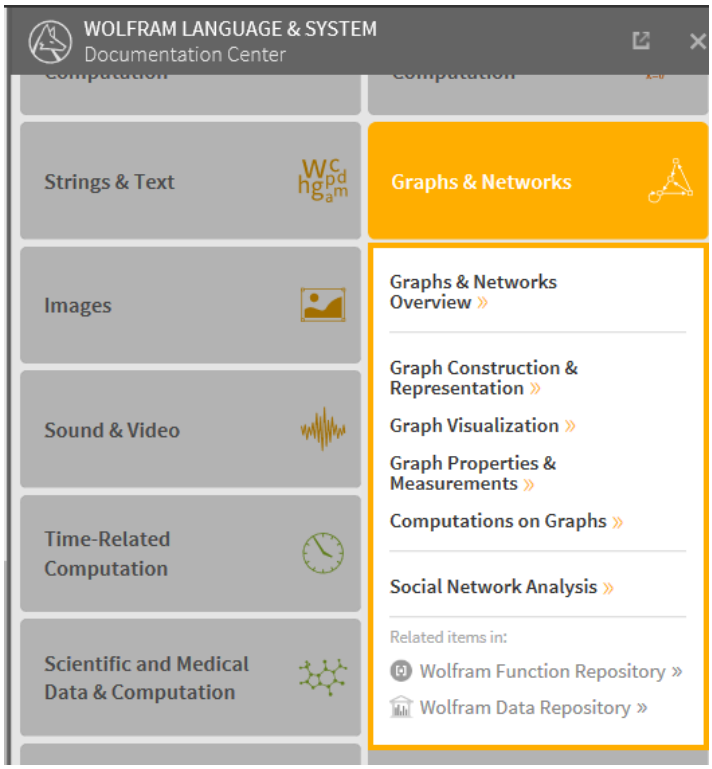


Выделить найденный подграф:

```
HighlightGraph[g, Subgraph[g, g1], GraphHighlightStyle → "Thick"]
```



Узнать принципы работы с любыми тремя функциями раздела Graphs&Networks справочной системы, объяснить работу.



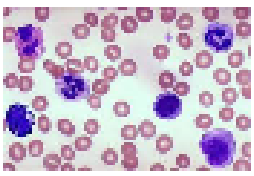
Задания к теме 3

1) Изучить примеры применения функций для обработки изображений, приведенные ниже и выполнить эти действия с другими изображениями. Работа выполняется в системе Wolfram Mathematica или на сайте, выбрать бесплатный вариант работы в облаке, wolfram.com → Все сайты и открытые ресурсы → Wolfram Language Sandbox

1.1. Работа с изображением, полученным через микроскоп

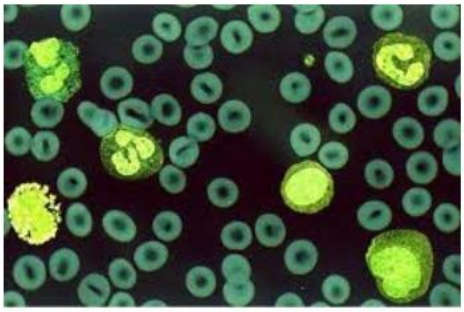
Импортируем изображение:

```
image =
```



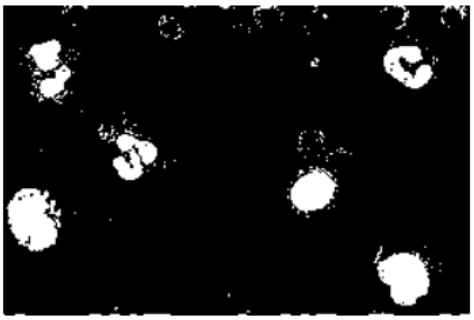
Формируем негатив:

```
image1 = ColorNegate[image]
```



Проводится бинаризация:

```
b = Binarize[image1, 0.6]
```



Выбираем связные области, удовлетворяющие заданным условиям:

```
d = SelectComponents[b, "Area", 500 < # < 9000 &]
```

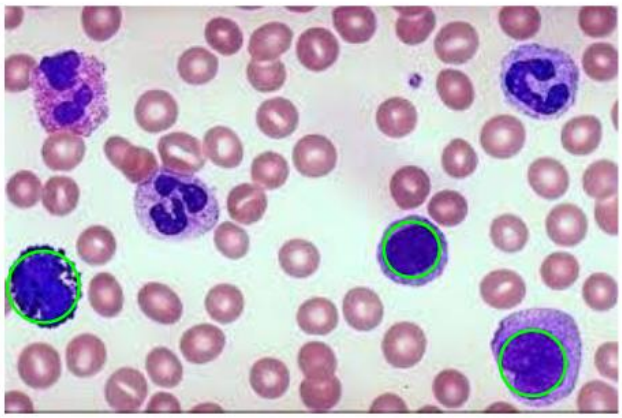
Накладываем на первоначальное изображение

```
e = ImageMultiply[image, d]
```

Находим координаты центра и значения радиусов кругов, соответствующих выделенным областям, выделяем области окружностями зеленого цвета.

```
g = ComponentMeasurements[e, {"Centroid", "EquivalentDiskRadius"}][[All, 2]]
{{{183.294, 73.4758}, 12.5524}, {{17.7718, 56.7042}, 16.0868}, {{237.589, 22.4913}, 14.777}}

Show[image, Graphics[{Green, Thick, Circle @@ #& /@ g}]]
```

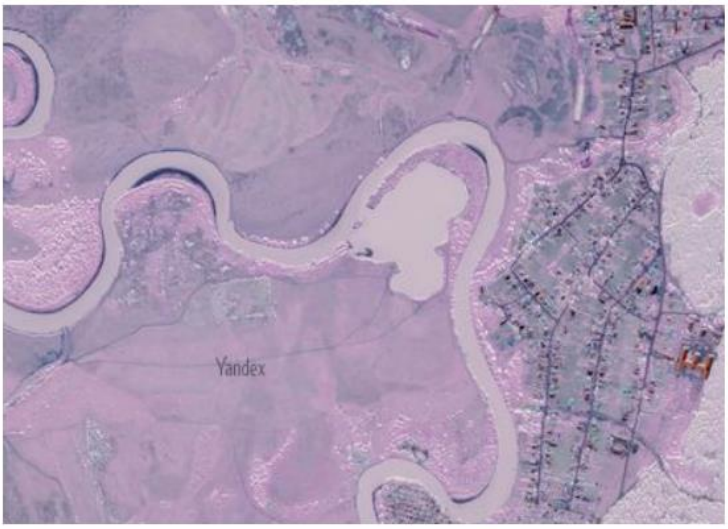


1.2. Работа с картой местности

Возьмём изображение местности на карте:

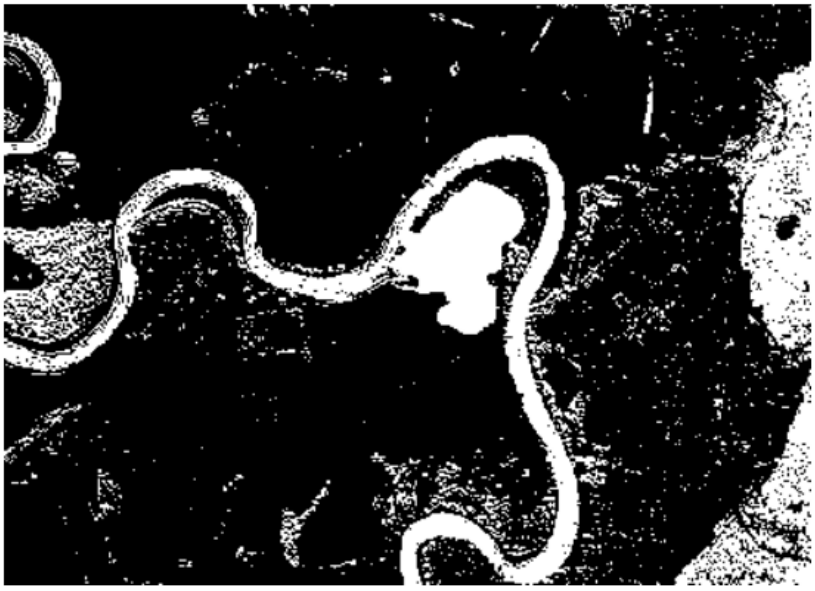
```
image =
```

```
a1 = ColorNegate[image]
```



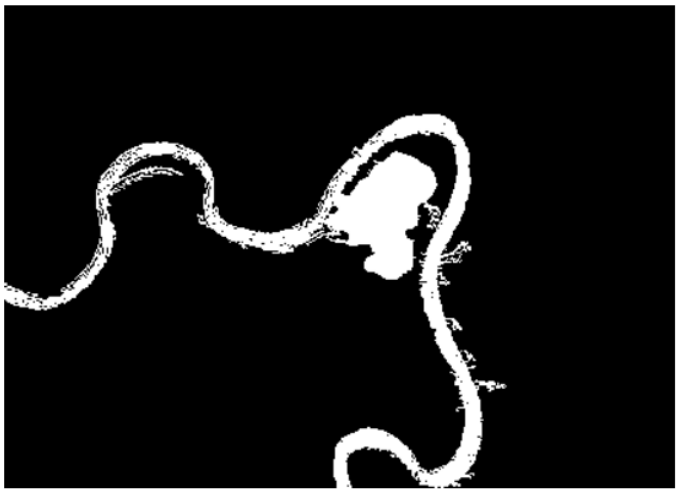
Применим функции MorphologicalComponents

```
a2 = MorphologicalComponents[a1, 0.7];
a3 = Image[a2]
```

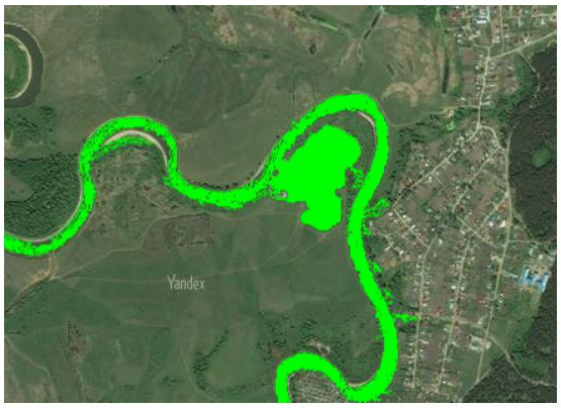


и SelectComponents, выберем наибольшую по площади связную область:

```
a4 = SelectComponents[a2, "Length", -1];
a5 = Image[a4]
```



Исследуем параметры, изменяем цвета и накладываем рисунки друг на друга. В результате получаем выделенную указанным цветом (в данном случае зеленым) область на карте.

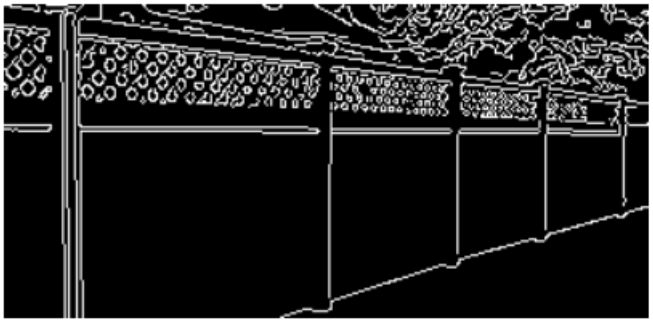


1.3. Ключевые точки изображения

Возьмем следующее изображение. Применим команду EdgeDetect и наложим изображения:

```
image =
```

```
a = EdgeDetect[image]
```



```
ImageAdd[image, a]
```

Для нахождения массива ключевых точек можно применить команду ImageKeypoints (показана часть массива):

```
b = ImageKeypoints[image, "Position"]
```

```
{{11.8513, 114.785}, {11.6663, 130.336}, {215.849, 126.619},
 {224.172, 121.919}, {308.299, 107.273}, {268.547, 112.37},
 {157.308, 125.932}, {268.607, 112.235}, {248.713, 136.012},
 {262.708, 36.3866}, {288.429, 118.218}, {11.7034, 123.013},
 {258.718, 131.327}, {205.158, 128.424}, {175.785, 126.254},
 {183.157, 131.631}, {269.773, 129.053}, {124.689, 134.79},
 {265.612, 107.343}, {195.318, 124.773}, {130.332, 139.697},
 {176.213, 124.105}, {233.254, 105.124}, {64.5295, 119.396},
 {277.851, 102.721}, {78.7975, 140.616}, {95.2999, 139.095},
```

```
Show[Rasterize[image, "Graphics"], Graphics[Point[b]]]
```

2) Работа выполняется в системе Wolfram Mathematica или на сайте, выбрать бесплатный вариант работы в облаке, wolfram.com → Все сайты и открытые ресурсы → Wolfram Language Sandbox

Работа с изображениями (раздел Image Processing & Analysis или Images).

1) Изучить раздел справочной системы Wolfram Mathematica

DATA MANIPULATION → Image Processing.

2) Выполнить импорт любых двух цветных фотоизображений (Import) . Путь к файлу указать с помощью пункта меню Insert → File Path …

а) с помощью команды ImageTake сделать их одинакового размера.

б) добавить рамку к первому изображению (ImagePad)

в) повернуть первое изображение на 90° (ImageRotate)

г) перевести первое изображение в черно-белое (ColorConvert)

д) найти края (границы, линии) в первом изображении (EdgeDetect)

е) разбить второе изображение на квадраты 8x8 пикселей (ImagePartition)

ж) наложить на второе изображение надпись Mathematica в виде рисунка ближе к левому нижнему углу (ImageCompose)

з) объединить два изображения в одно (ImageAssemble)

3) Подготовиться объяснить работу любых 8 команд (функций) раздела Image Processing, не использованных в пункте 2.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) А. Иванов, Д. Ильютко, Г. Носовский, А. Тужилин, А. Фоменко. Практикум по компьютерной геометрии. НОУ «ИНТУИТ». https://intuit.ru/studies/courses/645/501/info

2) Е. Седов. Основы работы в системе компьютерной алгебры Mathematica. НОУ «ИНТУИТ». https://intuit.ru/studies/courses/4765/1039/info

3) Интерактивные примеры: сайт Wolfram Research https://demonstrations.wolfram.com/?fp=right

4) Обзор возможностей Wolfram Language. https://reference.wolfram.com/language/

5) Шпаков, П. С. Основы компьютерной графики : учебное пособие / П. С. Шпаков, Ю. Л. Юнаков, М. В. Шпакова. Красноярск : СФУ, 2014, 398 с. Текст: электронный. URL: https://znanium.com/catalog/product/507976.

6) Н.Костюкова. Графы и их применение. НОУ «ИНТУИТ». https://intuit.ru/studies/courses/58/58/info

7) Бурзалова Т.В. Учебно-методический комплекс по решению задач дискретной математики с использованием компьютерной системы «Mathematica». Улан-Удэ: Издательство Бурятского госуниверситета, 2002 https://www.bsu.ru/content/page/1416/13.pdf

8) А. Куликов, Т. Овчинникова. Алгоритмические основы современной компьютерной графики. НОУ «ИНТУИТ». https://intuit.ru/studies/courses/70/70/info

9) Галимянов А.Ф., Миннегалиева Ч.Б. Обучение специальным приемам в системе Mathematica. Казань, изд-во КФУ, 2014. 52 с.