

2018

1

МОДЕЛИРОВАНИЕ И АНАЛИЗ ДАННЫХ

НАУЧНЫЙ ЖУРНАЛ

МОДЕЛИРОВАНИЕ И АНАЛИЗ ДАННЫХ

2018

ISSN 2219-3758

МОДЕЛИРОВАНИЕ И АНАЛИЗ ДАННЫХ

MODELLING AND DATA ANALYSIS

РЕДАКЦИОННАЯ КОЛЛЕГИЯ

Главный редактор – Л.С. Куравский

Заместители главного редактора – С.Д. Кулик, А.В. Пантелеев

Члены редакционной коллегии – К.К. Абгарян, Г.Г. Амосов, М.В. Воронов, Е.Л. Григоренко (*США*), В.К. Захаров, А.И. Кибзун, Л.М. Либкин (*Великобритания*), В.Р. Милов, А.В. Наумов, Д.Л. Ревизников, Х. Холлинг (*Германия*), Д. Фрэнсис (*США*), К.В. Хорошенков (*Великобритания*), Г.А. Юрьев

РЕДАКЦИОННЫЙ СОВЕТ

Председатель редакционного совета – Г.Г. Амосов

Члены редакционного совета – В.А. Барабанщиков, П. Бентлер (*США*), А.В. Горбатов, Л.С. Куравский, Л.М. Либкин (*Великобритания*), А.А. Марголис, В.В. Рубцов, Д.В. Ушаков, Д. Фрэнсис (*США*)

Ответственный секретарь – Н.Е. Юрьева

Издаётся с 2011 года

Учредитель

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный психолого-педагогический университет»

Адрес редколлегии:

г. Москва, ул. Сретенка, 29, факультет информационных технологий

Тел.: +7 (499) 167-66-74

E-mail: mad.mgppu@gmail.com

Журнал зарегистрирован в Государственном комитете РФ по печати.

Свидетельство о регистрации средств массовой информации

ПИ № ФС77-52058 от 7 декабря 2012 года

ISSN 2219-3758

© ФГБОУ ВО «Московский государственный психолого-педагогический университет», 2018

Все права защищены. Любая часть этого издания не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения редакционной коллегии.

Правила оформления рукописей, направляемых в редакцию журнала, высылаются по запросу по электронной почте.

СОДЕРЖАНИЕ CONTENTS

Компьютерные технологии как средство приобщения учащегося к математической реальности М.Е. Степанов Computer technologies as a means of familiarizing the student to the mathematical reality M.E. Stepanov	3
.....	
Использование игры PLines для психологического тестирования В.К. Войтов Using the game PLines for psychological testing V.K. Voitov	39
.....	
Преобразователь дискретной информации и пустое слово Й. Бокр Discrete information Converter and empty word J. Bokr	52
.....	
АВТОРЫ AUTHORS	60

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ КАК СРЕДСТВО ПРИБОЩЕНИЯ УЧАЩЕГОСЯ К МАТЕМАТИЧЕСКОЙ РЕАЛЬНОСТИ

М. Е. Степанов

В статье обсуждаются вопросы методики преподавания высшей математики, возникающие при современном уровне образования в нашей стране. Автор опирается на опыт работы на факультете информационных технологий МГППУ.

The article discusses the methods of teaching higher mathematics, arising at the modern level of education in our country. The author relies on his experience working at the Faculty of Information Technologies of the Moscow State University of Psychology and Education.

КЛЮЧЕВЫЕ СЛОВА

Высшее образование, методика преподавания математики, аналитическая геометрия, математический анализ, дифференциальная геометрия, линейная алгебра, алгебра многочленов, дифференциальные уравнения, уравнения математической физики, теория вероятностей, вычислительная математика, линейное программирование, компьютерные технологии.

1. ВВЕДЕНИЕ

Данная статья по содержанию примыкает к предыдущей статье автора «Некоторые вопросы методики преподавания высшей математики» [1] и развивает точку зрения, которая была там высказана: «При рассмотрении содержания математики, принято говорить о математическом понятии, вне зависимости от того, идёт ли речь о творческом развитии математического знания, об обучении математике или же о разработке методик преподавания. Нам, однако, более обоснованным представляется использование термина «объект математической реальности». Его введение позволяет детализировать процесс обучения и разделить его на формирование представления об объекте математической реальности как о понятии и как об образе. Только такое разделение позволяет достичь итогового соединения, условно говоря, логических аспектов владения объектом и аспектов интуитивного овладения им. Именно этот подход позволяет достигнуть того уровня видения математической реальности, которое называется пониманием».

В этой связи следует отметить, что развитие информатики привело к возникновению компьютерных технологий, которые оказались исключительно эффективным инструментом соединения логических аспектов познания объекта и аспектов интуитивного видения этого объекта и умения манипулировать им в виде пластического образа.

Автор отдаёт себе отчёт в том, что компьютерные технологии многогранны и предоставляют пользователю разнообразные средства опредмечивания математических объектов. Однако он сосредоточивает своё внимание на использовании для этой цели языков програм-

мирования. Если говорить более конкретно, речь идёт о языке Small Basic, разработанном в 2010 году специально под Windows. Конечно, нет оснований для отказа от использования в том же ключе и других языков программирования. Однако язык Small Basic освобождает от программистской формалистики и упрощает написание программ до предела.

Предлагаемые в дальнейшем подходы к обучению высшей математике опираются на использование языка программирования, как средства приобщения учащегося к математической реальности, и направлены на получение следующих результатов:

1. Студент должен увидеть, как теоретические положения и абстрактные формулы с помощью простой компьютерной программы превращаются в зримые образы.
2. Эти образы должны помочь студенту разобраться в смысле соответствующих математических понятий.
3. Студент должен научиться с достаточной степенью свободы преобразовывать математические идеи и формулы в алгоритмы соответствующих программ.

Ниже автор широко использует материалы из своей неизданной рукописи «Компьютерная геометрия, тт. 1 – 2).

2. ПРИМЕРЫ ЗАДАНИЙ

Растровая компьютерная графика основана на декартовой системе координат. По этой причине наиболее естественным исходным пунктом в деле использования компьютерных программ для иллюстрирования математических понятий является аналитическая геометрия на плоскости. Задания, которые можно предложить студенту на начальном этапе изучения аналитической геометрии, состоят в том, что нужно построить на экране некоторое изображение. При этом для его построения нужно произвести расчёты, основанные на хорошем понимании координатного метода. Когда соответствующая этому методу, пусть и простая, математика начинает работать, студент сразу ощущает, что речь идёт не об абстракциях, а о полезном и эффективном инструменте.

Первые изображения можно строить с помощью программ без переменных, в которых все операнды графических команд задаются числами. Далее необходим переход к написанию программ с переменными. И, как показывает опыт, для некоторых студентов этот шаг довольно некомфортен, поскольку думать о числах и думать о переменных – всё же разный уровень интеллектуального развития.

Конечной же целью заданий такого типа является освоение основ векторной алгебры и выход на должный уровень понимая аксиоматики Вейля, связывающей воедино точки и вектора. Дело в том, что студенты часто воспринимают точки и вектора порознь. Цель заданий на построение изображений состоит в том, чтобы донести до студента смысл точечно-векторного равенства типа $B = A + AB$.

Продемонстрируем одно из подобных заданий. Но предварим его инструкцией, которая предоставляется студентам и направлена на то, чтобы они поняли – написание соответствующей программы должно основываться не на смутных ощущениях, переходящих клавишами клавиатуры, а на математических расчётах.

Конечно, ряд студентов может произвести правильные и быстрые расчёты без упоминаемого ниже чертежа, но в настоящее время они, к сожалению, составляют меньшинство.

Кроме того, у студентов часто проявляется способность успешно проводить вычисления с конкретными числами, но переход к более абстрактным, но по содержанию совершенно аналогичным, вычислениям с переменными величинами ставит их в тупик. Конечно, такое положение дел является совершенно ненормальным. Предлагаемые нами задания на построение изображений должны помочь в устранении подобных недоработок школьной педагогики.

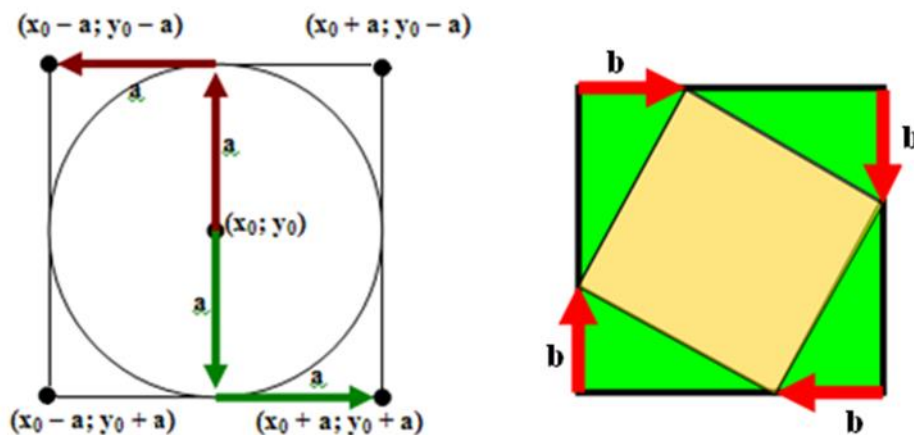
Инструкция по построению изображений на координатной плоскости.

1. Нарисуйте изображение на бумаге. Если можно, разбейте его на независимые части, каждую из которых можно рисовать, не обращая внимания на другие части полного изображения.
2. Выделите на рисунке опорные точки: концы отрезков; вершины прямоугольников; центры окружностей и эллипсов; обособленные точки.
3. Среди опорных точек выберите одну – исходную. Обозначьте её координаты через x и y .
4. Определите максимальный линейный размер изображения в пикселях, исходя из размеров экрана. Выберите конкретные значения для смещений по вертикали и горизонтали между опорными точками, а также значения радиусов.
5. Подберите конкретные значения координат x и y исходной точки, учитывая положение изображения на экране.
6. Вычислите, отправляясь от исходной точки, координаты всех опорных точек.
7. Определите порядок построения линий и фигур.
8. Напишите программу, наберите её и запустите.
9. В случае надобности займитесь поиском ошибок – отладкой программы.

Переходя к первому заданию ещё раз подчеркнём, что задания такого типа направлены на практическое освоение азов векторной алгебры. Кроме того, отметим, что некоторые задания мы будем сопровождать пояснениями, предназначенными для студентов.

Задание 1. Построить на экране наклонный квадрат.

Поскольку нет графического примитива, который строит прямоугольники со сторонами, имеющими негоризонтальное и невертикальное направления, нужно строить стороны квадрата с помощью примитивов-отрезков. Принцип расчёта координат, отправляясь от его центра, показан на чертеже.



Итоговая программа на языке Small Basic может выглядеть следующим образом.

```

x0 = 320
y0 = 220
a = 150
b = 50
x1 = x0 - a + b
y1 = y0 - a
x2 = x0 + a
y2 = y0 - a + b
x3 = x0 + a - b
y3 = y0 + a
x4 = x0 - a
y4 = y0 + a - b

```

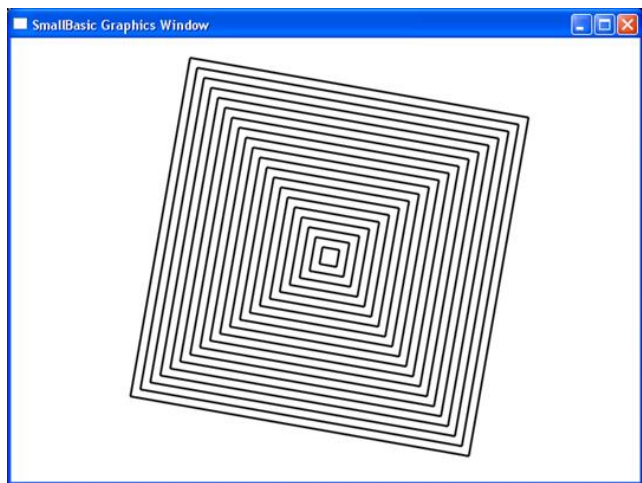
```
GraphicsWindow.DrawLine(x1,y1,x2,y2)
GraphicsWindow.DrawLine(x2,y2,x3,y3)
GraphicsWindow.DrawLine(x3,y3,x4,y4)
GraphicsWindow.DrawLine(x4,y4,x1,y1)
```

Отметим, что использование переменных позволяет в цикле строить семейства соответствующих фигур. При этом цикл вызывает изменение одной или нескольких переменных. В результате мы ведём подготовку к такому важнейшему понятию как преобразование, в данном случае речь идёт о преобразованиях геометрического характера.

Задание 2. Построить на экране семейство «концентрических» наклонных квадратов.

Вынесем все вычисления в подпрограмму. Для сохранения наклона квадратов введём коэффициент пропорциональности между переменными a и b

```
x0 = 320
y0 = 220
k = .3
For a = 10 To 200 Step 10
  b = a*k
  Наклонный_квадрат()
EndFor
Sub Наклонный_квадрат
  x1 = x0 - a + b
  y1 = y0 - a
  x2 = x0 + a
  y2 = y0 - a + b
  x3 = x0 + a - b
  y3 = y0 + a
  x4 = x0 - a
  y4 = y0 + a - b
  GraphicsWindow.DrawLine(x1,y1,x2,y2)
  GraphicsWindow.DrawLine(x2,y2,x3,y3)
  GraphicsWindow.DrawLine(x3,y3,x4,y4)
  GraphicsWindow.DrawLine(x4,y4,x1,y1)
EndSub
```



Нелишне на простом примере показать, что интерпретация вектора как сдвига вполне оправдана.

Задание 3. Построить ломаную по координатам вершин A, B, C, D . Сдвинуть её на заданный вектор. (Такой сдвиг называется параллельным, так как любой отрезок при переносе остаётся параллельным себе).

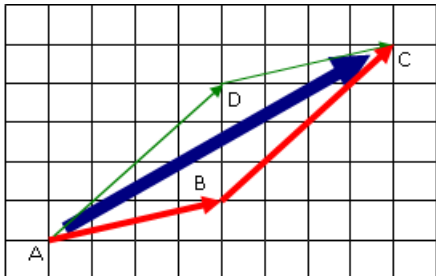
Пусть вектор сдвига равен $(50; 100)$, а вершины ломаной имеют координаты $A(100; 100)$, $B(200; 180)$, $C(300; 70)$ и $D(400; 150)$.

```
GraphicsWindow.DrawLine(100,100,200,180)
GraphicsWindow.DrawLine(200,180,300,70)
GraphicsWindow.DrawLine(300,70,400,150)
GraphicsWindow.PenColor="red"
GraphicsWindow.PenWidth=10
GraphicsWindow.DrawLine(100+50,100+100,200+50,180+100)
GraphicsWindow.DrawLine(200+50,180+100,300+50,70+100)
GraphicsWindow.DrawLine(300+50,70+100,400+50,150+100)
```



Пока мы делали упор на разложении векторов по двум направлениям. Теперь более внимательно рассмотрим операцию сложения векторов.

Поскольку смысл вектора состоит в сдвиге из точки в точку, легко понять, что сдвиг из точки А в точку В, а затем из В в точку С, приводит к сдвигу из точки А в точку С. И, кроме того, ясно, что если $AB(x_1; y_1)$ и $BC(x_2; y_2)$, то вектор АС имеет координаты $(x_1 + x_2; y_1 + y_2)$. Нами введена операция сложения векторов. Геометрическая запись для суммы такова:



$AC = AB + BC$, а алгебраическая запись имеет форму: $(x_1; y_1) + (x_2; y_2) = (x_1 + x_2; y_1 + y_2)$.

Задание 4. Сложение двух векторов с геометрической точки зрения производится по правилу параллелограмма. Ориентируясь на чертёж, сформулируйте это правило. На экране компьютера выполните следующие построения.

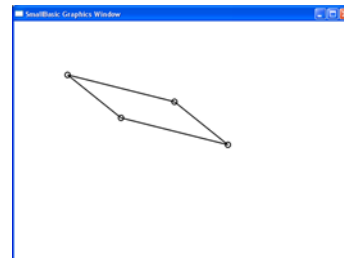
А) Заданы координаты точек А, В и D. Построить точку С и параллелограмм.

Б) Заданы координаты точек А, В и С. Построить точку D и параллелограмм.

Правило параллелограмма: сумма двух векторов АВ и AD является диагональю параллелограмма, сторонами которого являются отрезки АВ и AD. В координатах правило параллелограмма можно записать так: $x_c = x_a + x_{ab} + x_{ad}$, $y_c = y_a + y_{ab} + y_{ad}$.

Пункт А. Пусть А (100; 100), В (200; 180) и D (300; 150). Проведём предварительные вычисления, которые затем будут использованы в программе: $x_{ab} = 200 - 100 = 100$, $y_{ab} = 180 - 100 = 80$; $x_{ad} = 300 - 100 = 200$, $y_{ad} = 150 - 100 = 50$. Тогда $x_c = x_a + x_{ab} + x_{ad}$, $y_c = y_a + y_{ab} + y_{ad}$ или $x_c = 100 + 100 + 200 = 400$, $y_c = 100 + 80 + 50 = 230$. Программа:

```
GraphicsWindow.DrawEllipse (100 - 5, 100 - 5, 10,10)
GraphicsWindow.DrawEllipse (200 - 5, 180 - 5,10,10)
GraphicsWindow.DrawEllipse (300 - 5, 150 - 5,10,10)
GraphicsWindow.DrawEllipse (400 - 5, 230 - 5,10,10)
GraphicsWindow.DrawLine (100, 100, 200,180)
GraphicsWindow.DrawLine (200, 180,400,230)
GraphicsWindow.DrawLine (400, 230,300,150)
GraphicsWindow.DrawLine (300, 150,100,100)
```



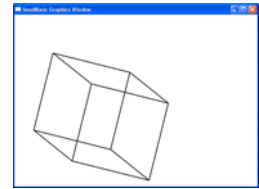
Пункт Б. Пусть А (100; 100), В (200; 180) и С (400; 230). Тогда $x_{bc} = 400 - 200 = 200$, $y_{bc} = 230 - 180 = 50$. Тогда $x_d = x_a + x_{bc}$, $y_d = y_a + y_{bc}$ или $x_d = 100 + 200 = 300$, $y_d = 100 + 50 = 150$. Программа аналогична предыдущей.

Задание 5. Постройте объёмное изображение параллелепипеда. Согласно Большой Советской Энциклопедии: «Параллелепипед (греч. *parallelepipedon*, от *parallelos* — параллельный и *epipedon* — плоскость), шестигранник, противоположные грани которого попарно параллельны. Параллелепипед имеет 8 вершин, 12 рёбер; его грани представляют собой попарно равные параллелограммы».

Зададим координаты трёх вершин параллелепипеда на его верхней грани: А (100; 100), В (200; 180) и D (300; 150). И, кроме того, сдвиг $(-50; 200)$, соответствующий боковому ребру параллелепипеда. Верхняя грань строится по пункту А предыдущего задания. Нижняя грань строится с помощью сдвига, как в задании 8. Наконец, должны быть построены и рёбра.


```

GraphicsWindow.DrawLine (100, 100, 200,180)
GraphicsWindow.DrawLine (200, 180,400,230)
GraphicsWindow.DrawLine (400, 230,300,150)
GraphicsWindow.DrawLine (300, 150,100,100)
GraphicsWindow.DrawLine (100 - 50, 100+200, 200 - 50,180+200)
GraphicsWindow.DrawLine (200 - 50, 180+200,400 - 50,230+200)
GraphicsWindow.DrawLine (400 - 50, 230+200,300 - 50,150+200)
GraphicsWindow.DrawLine (300 - 50, 150+200,100 - 50,100+200)
GraphicsWindow.DrawLine (100, 100, 100 - 50, 100+200)
GraphicsWindow.DrawLine (200, 180,200 - 50, 180+200)
GraphicsWindow.DrawLine (400, 230,400 - 50, 230+200)
GraphicsWindow.DrawLine (300, 150,300 - 50, 150+200)
    
```



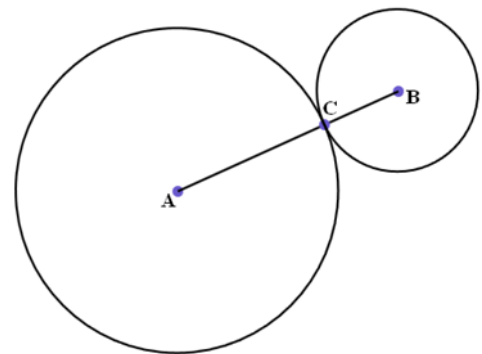
Задание 6. На плоскости заданы две точки $A(x_1; y_1)$ и $B(x_2; y_2)$. Построить третью точку C , которая делит отрезок AB так, что $\frac{AC}{CB} = 2$. По-

строить две окружности с центрами в точках A и B , проходящие через точку C .

В программе нужно одновременно и делить отрезок, и вычислять расстояния между точками.

```

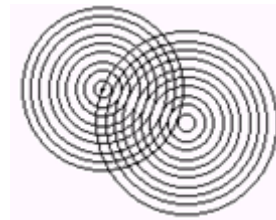
xa = 200
ya = 250
GraphicsWindow.FillEllipse(xa - 5, ya - 5, 10, 10)
xb = 400
yb = 150
GraphicsWindow.FillEllipse(xb - 5, yb - 5, 10, 10)
xc = xa + 2*(xb - xa)/3
yc = ya + 2*(yb - ya)/3
GraphicsWindow.DrawLine(xa, ya, xb, yb)
GraphicsWindow.FillEllipse(xc - 5, yc - 5, 10, 10)
ra = Math.Sqrt((xa - xc)*(xa - xc)+(ya - yc)*(ya - yc))
rb = Math.Sqrt((xb - xc)*(xb - xc)+(yb - yc)*(yb - yc))
GraphicsWindow.DrawEllipse(xa - ra, ya - ra, 2*ra, 2*ra)
GraphicsWindow.DrawEllipse(xb - rb, yb - rb, 2*rb, 2*rb)
    
```



Задание 7. На плоскости заданы две точки A и B . Построить два семейства концентрических окружностей с центрами в этих точках, так чтобы каждая окружность одного семейства касалась одной из окружностей другого семейства.

```

xa = 200
ya = 250
GraphicsWindow.FillEllipse(xa - 5, ya - 5, 10, 10)
xb = 400
yb = 150
GraphicsWindow.FillEllipse(xb - 5, yb - 5, 10, 10)
For lmb = 0 To 1 Step 1/10
xc = xa + lmb*(xb - xa)
yc = ya + lmb*(yb - ya)
GraphicsWindow.DrawLine(xa, ya, xb, yb)
GraphicsWindow.FillEllipse(xc - 5, yc - 5, 10, 10)
ra = Math.Sqrt((xa - xc)*(xa - xc)+(ya - yc)*(ya - yc))
rb = Math.Sqrt((xb - xc)*(xb - xc)+(yb - yc)*(yb - yc))
    
```



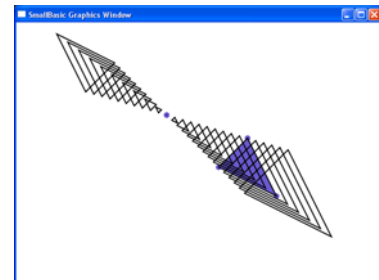
```
GraphicsWindow.DrawEllipse(xa - ra, ya - ra, 2*ra, 2*ra)
GraphicsWindow.DrawEllipse(xb - rb, yb - rb, 2*rb, 2*rb)
EndFor
```

Задание 8. С помощью параметрических уравнений прямой постройте пунктирный отрезок.

```
x0 = 50
y0 = 100
GraphicsWindow.FillEllipse(x0 - 5, y0 - 5, 10, 10)
x1 = 600
y1 = 400
GraphicsWindow.FillEllipse(x1 - 5, y1 - 5, 10, 10)
For lmb = 0 To 1 Step 1/100
  x = x0 + lmb*(x1 - x0)
  y = y0 + lmb*(y1 - y0)
  GraphicsWindow.SetPixel(x, y, "")
EndFor
```

Задание 9. Постройте семейство гомотетичных треугольников.

```
x0 = 260
y0 = 160
GraphicsWindow.FillEllipse(x0 - 5, y0 - 5, 10, 10)
xa = 400
ya = 200
GraphicsWindow.FillEllipse(xa - 5, ya - 5, 10, 10)
xb = 450
yb = 300
GraphicsWindow.FillEllipse(xb - 5, yb - 5, 10, 10)
xc = 350
yc = 250
GraphicsWindow.FillEllipse(xc - 5, yc - 5, 10, 10)
GraphicsWindow.FillTriangle(xa, ya, xb, yb, xc, yc)
For lmb = -1 To 1.5 Step 1/10
  x1 = x0 + lmb*(xa - x0)
  y1 = y0 + lmb*(ya - y0)
  x2 = x0 + lmb*(xb - x0)
  y2 = y0 + lmb*(yb - y0)
  x3 = x0 + lmb*(xc - x0)
  y3 = y0 + lmb*(yc - y0)
  GraphicsWindow.DrawTriangle(x1, y1, x2, y2, x3, y3)
EndFor
```



Задание 10. С помощью предыдущей программы создайте анимационный эффект, демонстрирующий гомотетию в динамике.

Для создания анимационного эффекта нужно уменьшить шаг в цикле до одной сотой и перед EndFor вставить три строки, которые осуществляют задержку кадра для фиксации глазом и очистку экрана:

```
For tm = 1 To 50000
EndFor
GraphicsWindow.Clear()
```

Приведём ещё три задания, направленных на освоение осознанной работы с параметрическими уравнениями прямой, расстояниями и отношениями длин отрезков.

Задание 11. На плоскости заданы две точки А и В. Построить на прямой АВ точку А₁, такую, чтобы длина отрезка АА₁ равнялась 100 пикселям. Подтвердить результат, построив окружность с центром в точке А и радиусом 100.

```

ха = 200
ya = 250
GraphicsWindow.FillEllipse(ха - 5, ya - 5, 10, 10)
xb = 400
yb = 150
GraphicsWindow.FillEllipse(xb - 5, yb - 5, 10, 10)
GraphicsWindow.DrawLine(ха, ya, xb, yb)
dl = Math.Sqrt((ха - xb)*(ха - xb) + (ya - yb)*(ya - yb))
lmb = 100/dl
xc = ха + lmb*(xb - ха)
yc = ya + lmb*(yb - ya)
GraphicsWindow.FillEllipse(xc - 5, yc - 5, 10, 10)
GraphicsWindow.DrawEllipse(ха - 100, ya - 100, 200, 200)

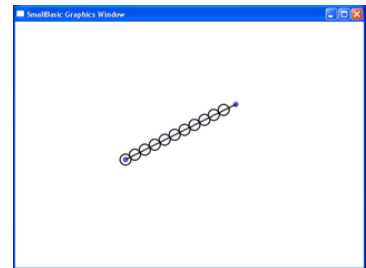
```

Задание 12. На плоскости заданы две точки А и В. Построить на прямой АВ несколько точек А₁, А₂, А₃ ... таких, что АА₁ = А₁А₂ = А₂А₃ = ... = 20.

```

ха = 200
ya = 250
GraphicsWindow.FillEllipse(ха - 5, ya - 5, 10, 10)
xb = 400
yb = 150
GraphicsWindow.FillEllipse(xb - 5, yb - 5, 10, 10)
GraphicsWindow.DrawLine(ха, ya, xb, yb)
dl = Math.Sqrt((ха - xb)*(ха - xb) + (ya - yb)*(ya - yb))
lmb = 20/dl
For n = 0 To 10
  xc = ха + n*lmb*(xb - ха)
  yc = ya + n*lmb*(yb - ya)
  GraphicsWindow.FillEllipse(xc - 2, yc - 2, 4, 4)
  GraphicsWindow.DrawEllipse(xc - 10, yc - 10, 20, 20)
EndFor

```

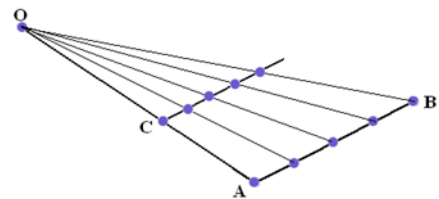


Задание 13. На плоскости заданы отрезок ОА и точка В. Выбрать на отрезке ОА точку С, провести прямую АВ и параллельную ей линию через точку С. Построить равномерную последовательность точек на АВ, а также их центральные проекции из точки О на вторую прямую.

```

хо = 50
yo = 50
GraphicsWindow.FillEllipse(хо - 5, yo - 5, 10, 10)
ха = 320
ya = 400
GraphicsWindow.FillEllipse(ха - 5, ya - 5, 10, 10)

```



```
yb = 200
GraphicsWindow.FillEllipse(xb - 5, yb - 5, 10, 10)
GraphicsWindow.DrawLine(xo, yo, xa, ya)
GraphicsWindow.DrawLine(xb, yb, xa, ya)
lmbc = 2/3
xc = xo + lmbc*(xa - xo)
yc = yo + lmbc*(ya - yo)
GraphicsWindow.FillEllipse(xc - 5, yc - 5, 10, 10)
xd = xo + lmbc*(xb - xo)
yd = yo + lmbc*(yb - yo)
GraphicsWindow.DrawLine(xc, yc, xd, yd)
For lmbc = 0 To 1 Step 1/20
  xx = xa + lmbc*(xb - xa)
  yy = ya + lmbc*(yb - ya)
  GraphicsWindow.FillEllipse(xx - 5, yy - 5, 10, 10)
  GraphicsWindow.DrawLine(xo, yo, xx, yy)
  xx = xc + lmbc*(xd - xc)
  yy = yc + lmbc*(yd - yc)
  GraphicsWindow.FillEllipse(xx - 5, yy - 5, 10, 10)
EndFor
```

Отметим, что задачи на построение компьютерных чертежей, относящихся к элементарной планиметрии, требуют применения методов аналитической геометрии.

Задание 14. Постройте медианы треугольника, выделите маленькими окружностями вершины треугольника, середины сторон и точку пересечения медиан.

```
xa = 200
ya = 300
xb = 400
yb = 100
xc = 500
yc = 350
GraphicsWindow.DrawTriangle(xa, ya, xb, yb, xc, yc)
xma = (xb + xc)/2
yma = (yb + yc)/2
GraphicsWindow.FillEllipse(xma - 5, yma - 5, 10, 10)
GraphicsWindow.DrawLine(xa, ya, xma, yma)
xmb = (xa + xc)/2
ymb = (ya + yc)/2
GraphicsWindow.FillEllipse(xmb - 5, ymb - 5, 10, 10)
GraphicsWindow.DrawLine(xb, yb, xmb, ymb)
xmc = (xb + xa)/2
ymc = (yb + ya)/2
GraphicsWindow.FillEllipse(xmc - 5, ymc - 5, 10, 10)
GraphicsWindow.DrawLine(xc, yc, xmc, ymc)
xmed = xma + (xa - xma)/3
ymed = yma + (ya - yma)/3
GraphicsWindow.FillEllipse(xmed - 5, ymed - 5, 10, 10)
```

Задание 15. Теорема о биссектрисе треугольника гласит: *биссектриса AP угла A треугольника ABC делит сторону BC на отрезки BP и PC, такие что $BP : PC = AB : AC$* . Ис-



пользуя этот факт, постройте биссектрисы треугольника, выделите маленькими окружностями вершины треугольника, точки пересечения биссектрис со сторонами и точку пересечения биссектрис.

```
xa = 200
ya = 300
xb = 400
yb = 100
xc = 500
yc = 350
GraphicsWindow.DrawTriangle(xa, ya, xb, yb, xc, yc)
a = Math.Sqrt((xb - xc)*(xb - xc)+(yb - yc)*(yb - yc))
b = Math.Sqrt((xa - xc)*(xa - xc)+(ya - yc)*(ya - yc))
c = Math.Sqrt((xb - xa)*(xb - xa)+(yb - ya)*(yb - ya))
xa1 = xb + c*(xc - xb)/(b+c)
ya1 = yb + c*(yc - yb)/(b+c)
GraphicsWindow.FillEllipse(xa1 - 5, ya1 - 5, 10, 10)
GraphicsWindow.DrawLine(xa, ya, xa1, ya1)
xb1 = xc + a*(xa - xc)/(a+c)
yb1 = yc + a*(ya - yc)/(a+c)
GraphicsWindow.FillEllipse(xb1 - 5, yb1 - 5, 10, 10)
GraphicsWindow.DrawLine(xb, yb, xb1, yb1)
xc1 = xa + b*(xb - xa)/(b+a)
yc1 = ya + b*(yb - ya)/(b+a)
GraphicsWindow.FillEllipse(xc1 - 5, yc1 - 5, 10, 10)
GraphicsWindow.DrawLine(xc, yc, xc1, yc1)
otr = Math.Sqrt((xb - xa1)*(xb - xa1)+(yb - ya1)*(yb - ya1))
xbis = xa1 + otr*(xa - xa1)/(c+otr)
ybis = ya1 + otr*(ya - ya1)/(c+otr)
GraphicsWindow.FillEllipse(xbis - 5, ybis - 5, 10, 10)
```

Конечно, все приведённые выше задания элементарны, однако не следует забывать, что они используются для того, чтобы студент мог свободно пользоваться исходными понятиями аналитической геометрии. Кроме того, задания в предлагаемой форме перестают быть школярскими упражнениями, а превращаются, пусть и в простую, на инженерную задачу. При этом теоретические представления опредмечиваются и воплощаются в зримый образ.

Следующим этапом освоения аналитической геометрии после изучения метода координат является знакомство с кривыми, описываемыми алгебраическими уравнениями первого и второго порядка.

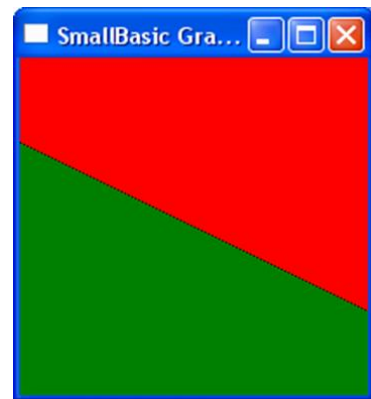
При этом возникает проблема масштабирования, состоящая в следующем. Экранное пространство имеет координатное поле с жёстко заданными координатами. При построении кривых нужно гибко менять характер системы координат, приравливая её к математическим нуждам. Вопрос решается достаточно просто. Векторное уравнение прямой $X = A + \lambda \cdot AB$ ставит в соответствие точке X , имеющей некоторую экранную координату, теоретическую координату λ . Таким образом, при построении алгебраических кривых можно поступать следующим образом. Выберем на экране точку $(x_0; y_0)$ в качестве начала координат. Предположим, что из неё исходят два вектора – горизонтальный и вертикальный. При этом вектора имеют одинаковую длину a (в пикселях). Тогда экранные координаты точки M , соответствующей теоретическим координатам $(t; u)$ вычисляются по формулам: $x_e = x_0 + t \cdot a$, $y_e = y_0 - u \cdot a$.

Наличие знака минус во второй формуле связано с тем, что в экранной системе координат ось ординат направлена вниз, и мы должны её перевернуть в привычное положение.

Как известно, прямая, проходящая через точку $(x_0; y_0)$ перпендикулярно вектору $(a; b)$, имеет уравнение $a \cdot (x - x_0) + b \cdot (y - y_0) = 0$. Эта прямая делит плоскость на две полуплоскости. Если в уравнение прямой подставить координаты точки, не лежащей на прямой, то для точек одной из упомянутых полуплоскостей значение всегда будет больше нуля, а для другой – меньше. Аналогичная ситуация имеет место и для любого неявного уравнения $F(x, y) = 0$, описывающего некоторую кривую. Причины этого обстоятельства нетрудно объяснить. Пусть нам задано неравенство $F(x_1, x_2) \leq 0$. Рассмотрим функцию от двух переменных $z = F(x_1, x_2)$. Она задаёт поверхность в трёхмерном пространстве. Пересечение этой поверхности с плоскостью $z = 0$ является некоторой кривой (её уравнение $F(x_1, x_2) = 0$), разбивающей плоскость на две области. В одной области $F(x_1, x_2) \leq 0$, а в другой $F(x_1, x_2) \geq 0$. Итак, неравенство $F(x_1, x_2) \leq 0$ задаёт на плоскости область, ограниченную кривой $F(x_1, x_2) = 0$. Используя этот факт, мы можем, пусть и неэкономным образом, строить на экране кривые. Более того, мы одновременно получаем геометрическое решение неравенств вида $F(x_1, x_2) \leq 0$. Способ построения таков. Сканируем экранное пространство с помощью вложенного цикла по теоретическим переменным t и u и вычисляем значение функции $z = F(x_1, x_2)$. Если z больше нуля, ставим точку одним цветом, в противном случае – другим.

Задание 16. Построить на экране прямую вида $ax + by + c = 0$ и решить соответствующие неравенства геометрически.

```
GraphicsWindow.Width = 600
GraphicsWindow.Height = 600
x0 = 300
y0 = 300
ed = 100
a = 2
b = 3
c = 1
For t = -3 To 3 Step .01
  For u = -3 To 3 Step .01
    xe = x0 + ed*t
    ye = y0 - ed*u
    If a*t + b*u + c < 0 Then
      GraphicsWindow.SetPixel(xe,ye,"green")
    Else
      GraphicsWindow.SetPixel(xe,ye,"red")
    EndIf
  EndFor
EndFor
EndFor
```

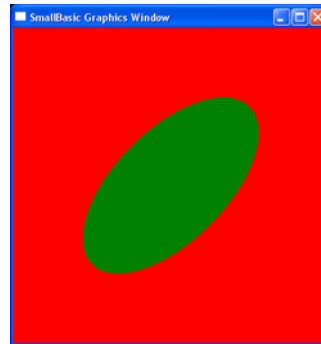


Задание 17. Построить на экране кривую вида $ax^2 + bxy + cy^2 + d = 0$ и решить соответствующие неравенства геометрически.

```

GraphicsWindow.Width = 400
GraphicsWindow.Height = 400
x0 = 200
y0 = 200
ed = 100
a = 5
b = -6
c = 5
d = -4
For t = -2 To 2 Step .01
  For u = -2 To 2 Step .01
    xe = x0 + ed*t
    ye = y0 - ed*u
    If a*t*t + b*t*u + c*u*u + d < 0 Then
      GraphicsWindow.SetPixel(xe,ye,"green")
    Else
      GraphicsWindow.SetPixel(xe,ye,"red")
    EndIf
  EndFor
EndFor
EndFor

```



Важной составляющей частью аналитической геометрии на плоскости является раздел, изучающий прямые линии. Здесь рассматриваются различные виды уравнений, описывающих прямые, и изучаются вопросы, связанные с пересечением прямых, с определением углов между прямыми и вычислением расстояний от прямых до точек. Следует отметить, что, казалось бы несложный вопрос о вычислении координат точки пересечения двух прямых, органично связывает аналитическую геометрию с проективной.

Перечислим основные виды уравнений, описывающих прямые. На их основе могут быть предложены многочисленные задания.

1. Параметрические уравнения прямой: $x = x_A + \lambda \cdot p$ и $y = y_A + \lambda \cdot q$. При построении прямой в цикле изменяется параметр λ . Точка $(x_A; y_A)$ лежит на прямой, вектор $(p; q)$ указывает направление прямой.

2. Если задана точка $(x_A; y_A)$ и угол наклона u прямой, то $p = \cos u$ и $q = \sin u$.

3. Каноническое уравнение прямой: $y = k \cdot x + b$. При построении прямой в цикле изменяется независимый аргумент x . Коэффициент k является тангенсом угла наклона прямой.

4. Если задана точка $(x_0; y_0)$ через которую проходит прямая и тангенс её угла наклона k , то уравнение прямой принимает вид $y = k \cdot x + y_0 - k \cdot x_0$.

5. Если заданы две точки $(x_0; y_0)$ и $(x_1; y_1)$, через которые проходит прямая, то тангенс угла наклона прямой равен $k = \frac{y_1 - y_0}{x_1 - x_0}$.

6. Прямые с уравнениями $y = k \cdot x + b_1$ и $y = k \cdot x + b_2$ параллельны.

7. Если тангенс угла наклона исходной прямой равен k , то тангенс угла наклона прямой, перпендикулярной к ней, равен $k_1 = -\frac{1}{k}$.

8. Уравнение прямой общего вида: $a \cdot (x - x_0) + b \cdot (y - y_0) = 0$ или $ax + by + c = 0$, где $c = -x_0 \cdot a - y_0 \cdot b$. Это уравнение непосредственно для построения прямой не используется. Оно требует предварительного преобразования, после которого либо y выражается через x ,

либо x – через y . Уравнение содержит в себе возможность описания горизонтальных ($y = y_0$) и вертикальных ($x = x_0$) прямых. Анализ уравнения может проводиться непосредственно в программе с помощью условных операторов. Прямая, описываемая уравнением $a \cdot (x - x_0) + b \cdot (y - y_0) = 0$, проходит через точку (x_0, y_0) перпендикулярно к вектору $(a; b)$.

Рассмотрим несколько заданий, математической основой которых являются перечисленные нами уравнения. При комментировании для краткости будем ссылаться на номера пунктов, содержащих описание нужных уравнений.

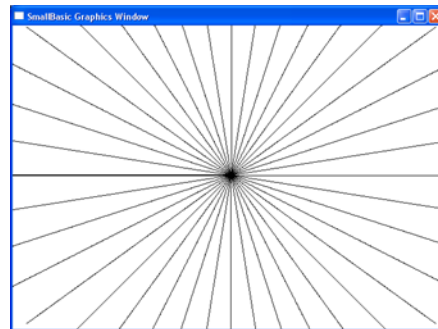
Задание 18. Множество прямых на плоскости, проходящих через одну точку, называется пучком прямых. Построить пучок с центром в начале координат.

При решении задачи используем пункт 2. Меняя в цикле угол, то есть, вращая направляющий вектор, поочередно строим прямые пучка. Для этого в свою очередь во внутреннем цикле меняем параметр λ .

```

pi= Math.Pi
x0 = 320
y0 = 220
ed=100
For u=0 To pi Step pi/20
  p= Math.Cos(u)
  q= Math.Sin(u)
  For lmb= -3.7 To 3.7 Step .01
    x = x0 + lmb*p*ed
    y = y0 - lmb*q*ed
    GraphicsWindow.SetPixel(x,y,"")
  EndFor
EndFor

```



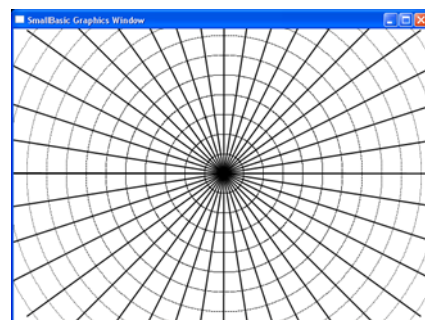
Интересен тот факт, что на уравнения $x = x_A + \lambda \cdot \cos u$ и $y = y_A + \lambda \cdot \sin u$ можно посмотреть двояким образом. Если изменяется параметр λ , перед нами параметрические уравнения прямой. Если же изменяется параметр u , перед нами параметрические уравнения окружности радиуса λ .

Задание 19. Построить пучок прямых с центром в начале координат и семейство концентрических окружностей с центром в центре пучка прямых.

```

pi= Math.Pi
x0 = 320
y0 = 220
ed=100
For u=0 To pi Step pi/20
  p= Math.Cos(u)
  q= Math.Sin(u)
  lmb = -3.7
  x1 = x0 + lmb*p*ed
  y1 = y0 - lmb*q*ed
  lmb = 3.7
  x2 = x0 + lmb*p*ed
  y2 = y0 - lmb*q*ed
  GraphicsWindow.DrawLine(x1,y1,x2,y2)
EndFor

```




```
For u=0 To 2*pi Step pi/500
  p= Math.Cos(u)
  q= Math.Sin(u)
  For lmb= 0 To 3.6 Step .3
    x = x0 + lmb*p*ed
    y = y0 - lmb*q*ed
    GraphicsWindow.SetPixel(x,y,"")
  EndFor
EndFor
```

Задание 20. Используя правило Крамера для решения систем уравнений построить точку пересечения двух прямых, заданных уравнениями $a_1 \cdot x + a_2 \cdot y + a_3 = 0$ и $b_1 \cdot x + b_2 \cdot y + b_3 = 0$.

Формулировка правила Крамера. Пусть задана система двух линейных уравнений с двумя неизвестными:

$$\begin{cases} a_1 \cdot x + a_2 \cdot y + a_3 = 0 \\ b_1 \cdot x + b_2 \cdot y + b_3 = 0 \end{cases}$$

Чтобы найти её решение, из матрицы коэффициентов поочередно вычёркивают первый, второй и третий столбцы, а оставшиеся не вычеркнутыми столбцы располагают в циклическом порядке (за тройкой следует единица). Для трёх полученных матриц вычисляют определители (из произведения элементов, стоящих на главной диагонали, вычитают произведение элементов, стоящих на побочной диагонали) $\Delta_1, \Delta_2, \Delta_3$. Тогда $x = \frac{\Delta_1}{\Delta_3}$ и $y = \frac{\Delta_2}{\Delta_3}$.

Теперь свяжем уравнения прямых с воображаемой плоскостью (координаты t и u). Для перехода к экранной системе координат необходимо указать начало системы координат $(x_0; y_0)$ и масштаб (ed) . Уравнения, описывающие прямые, можно привести к каноническому виду: $u = \frac{-a_1 t - a_3}{a_2}$ и $x = \frac{-b_1 t - b_3}{b_2}$. Прямые строятся в цикле с параметром t , пробегающим отрезок от -4 до 4 .

Наконец точка пересечения определяется по правилу Крамера: вычисляем три определителя и находим координаты t_{peresech} и u_{peresech} , а затем преобразуем их в экранные координаты.

```
x0 = 320
y0 = 220
ed = 100
a1 = 1
a2 = 1
a3 = 1
b1 = -1
b2 = 2
b3 = -3
For t = -4 To 4 Step .01
  u = (-a1*t - a3)/a2
  x = x0 + ed*t
  y = y0 - ed*u
  GraphicsWindow.SetPixel(x,y,"")
u = (-b1*t - b3)/b2
x = x0 + ed*t
```

```

y = y0 - ed*u
GraphicsWindow.SetPixel(x,y,"")
EndFor
delta1 = a2*b3 - a3*b2
delta2 = a3*b1 - a1*b3
delta3 = a1*b2 - a2*b1
tperesech = delta1/delta3
uperesech = delta2/delta3
x = x0 + ed*tperesech
y = y0 - ed*uperesech
GraphicsWindow.FillEllipse(x - 5,y - 5,10,10)

```

Поскольку в предыдущем задании для точки пересечения двух прямых фактически были найдены её проективные координаты, уместно будет рассмотреть принцип двойственности. Его суть состоит в том, что прямые и точки в проективной геометрии становятся как бы зеркальным отражением друг друга. Например, утверждение «через любые две точки проходит единственная прямая» после «отражения» превращается в утверждение «две любые прямые пересекаются в единственной точке».

Существуют и вычислительные аспекты принципа двойственности. Итак, положение точек определяется проективными координатами, то есть тройками чисел $(x_1; x_2; x_3)$, а уравнения прямых принимают форму: $a_1x_1 + a_2x_2 + a_3x_3 = 0$. При этом возврат к исходным координатам и уравнениям весьма прост. Положив $\tilde{\delta} = \frac{\tilde{\delta}_1}{\tilde{\delta}_3}$ и $\tilde{\sigma} = \frac{\tilde{\delta}_2}{\tilde{\delta}_3}$, приходим к от уравнения

$a_1x_1 + a_2x_2 + a_3x_3 = 0$ к уравнению вида $a_1x + a_2y + a_3 = 0$.

Уравнение $a_1x_1 + a_2x_2 + a_3x_3 = 0$ прекрасно отражает принцип двойственности. Тройки чисел $(a_1; a_2; a_3)$ и $(x_1; x_2; x_3)$ занимают в нём симметричное положение. Это позволяет решить следующую задачу. На плоскости с помощью проективных координат заданы две точки $(x_1; x_2; x_3)$ и $(y_1; y_2; y_3)$. Найти уравнение прямой, проходящей через эти точки.

Ясно, что при подстановке координат точек в искомое уравнение, оно будет выполняться. По этой причине мы приходим к системе двух линейных уравнений с неизвестными $a_1; a_2$ и a_3 :

$$\begin{cases} a_1x_1 + a_2x_2 + a_3x_3 = 0 \\ a_1y_1 + a_2y_2 + a_3y_3 = 0. \end{cases}$$

Решения данной системы определяется следующими формулами:

$$a_1 = \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix}, a_2 = \begin{vmatrix} x_3 & x_1 \\ y_3 & y_1 \end{vmatrix}, a_3 = \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix}.$$

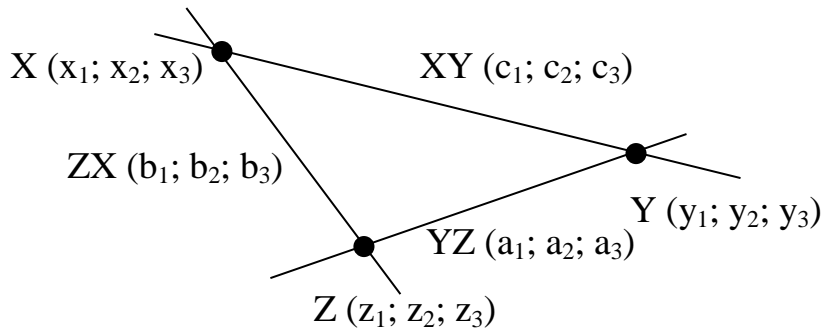
Таким образом, вычислительные процедуры для нахождения уравнения прямой, проходящей через две заданные точки и для нахождения координат точки пересечения двух прямых одинаковы. Коэффициенты $a_1; a_2$ и a_3 называют проективными координатами прямой.

Задание 21. Проверить идентичность описанных выше вычислительных процедур с помощью специальной программы.

Для проведения соответствующей проверки с помощью проективных координат зададим на плоскости три точки X, Y и Z. Затем найдём уравнения прямых XY, YZ и ZX, после чего вычислим координаты их точек пересечения и сравним с исходными координатами то-



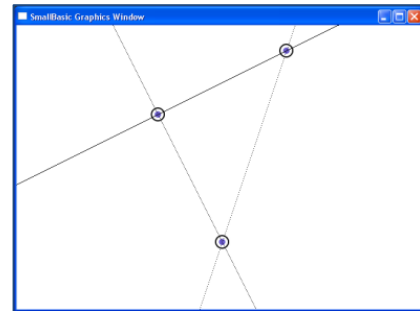
чек X, Y, Z . На чертеже показаны обозначения, которые мы даём проективным координатам соответствующих точек и прямых.



```

x0 = 320
y0 = 240
ed = 100
x1 = -1
x2 = 1
x3 = 1
GraphicsWindow.FillEllipse(x0+ed*(x1/x3) - 5,y0 - ed*(x2/x3) - 5,10,10)
y1 = 1
y2 = 2
y3 = 1
GraphicsWindow.FillEllipse(x0+ed*(y1/y3) - 5,y0 - ed*(y2/y3) - 5,10,10)
z1 = 0
z2 = -1
z3 = 1
GraphicsWindow.FillEllipse(x0+ed*(z1/z3) - 5,y0 - ed*(z2/z3) - 5,10,10)
a1 = y2*z3 - y3*z2
a2 = y3*z1 - y1*z3
a3 = y1*z2 - y2*z1
For t = -4 To 4 Step .01
  u = (-a1*t - a3)/a2
  GraphicsWindow.SetPixel(x0+ed*t,y0 - ed*u,"")
EndFor
b1 = x2*z3 - x3*z2
b2 = x3*z1 - x1*z3
b3 = x1*z2 - x2*z1
For t = -4 To 4 Step .01
  u = (-b1*t - b3)/b2
  GraphicsWindow.SetPixel(x0+ed*t,y0 - ed*u,"")
EndFor
c1 = x2*y3 - x3*y2
c2 = x3*y1 - x1*y3
c3 = x1*y2 - x2*y1
For t = -4 To 4 Step .01
  u = (-c1*t - c3)/c2
  GraphicsWindow.SetPixel(x0+ed*t,y0 - ed*u,"")
EndFor
x1 = b2*c3 - b3*c2
x2 = b3*c1 - b1*c3
x3 = b1*c2 - b2*c1

```

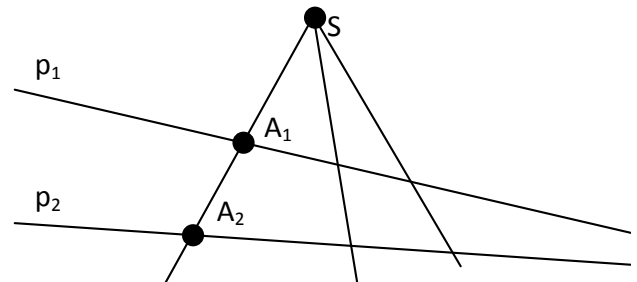


```

GraphicsWindow.DrawEllipse(x0+ed*(x1/x3) - 10,y0 - ed*(x2/x3) - 10,20,20)
y1 = a2*c3 - a3*c2
y2 = a3*c1 - a1*c3
y3 = a1*c2 - a2*c1
GraphicsWindow.DrawEllipse(x0+ed*(y1/y3) - 10,y0 - ed*(y2/y3) - 10,20,20)
z1 = a2*b3 - a3*b2
z2 = a3*b1 - a1*b3
z3 = a1*b2 - a2*b1
GraphicsWindow.DrawEllipse(x0+ed*(z1/z3) - 10,y0 - ed*(z2/z3) - 10,20,20)
    
```

Задание 22. На плоскости заданы точка S и две прямые p_1 и p_2 . Написать программу, которая позволит из центра S проективно отобразить прямую p_1 на прямую p_2 .

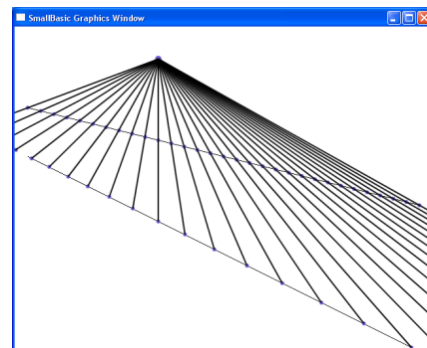
Пусть прямая p_1 задана точкой M с декартовыми координатами $(m_1; m_2)$ и вектором $(p; q)$. Тогда любая лежащая на ней точка A_1 имеет проективные координаты $(m_1 + \lambda \cdot p; m_2 + \lambda \cdot q; 1)$. Если параметр λ пробегает все действительные значения от минус до плюс бесконечности, то точка A_1 заметает всю прямую p_1 .



Таким образом, меняя параметр λ , мы фактически перебираем все прямые SA_1 , образующие пучок с центром в точке S . Чтобы решить задачу, нам необходимо найти координаты точки пересечения прямой p_2 с прямой SA_1 . Прямую p_2 зададим с помощью трёх коэффициентов c_1, c_2 и c_3 .

```

GraphicsWindow.Width=640
GraphicsWindow.Height=500
x0=320
y0=250
ed=100
s1=-1
s2=2
s3=1
xs=x0+s1*ed
ys=y0-s2*ed
GraphicsWindow.FillEllipse(xs - 5,ys - 5,10,10)
m1= -2
m2= 1
p= 1
q= -.25
For lmb= -1 To 5 Step .01
    t1=m1+lmb*p
    t2=m2+lmb*q
    x=x0+t1*ed
    y=y0 - t2*ed
    GraphicsWindow.SetPixel(x,y,"")
EndFor
c1=1
c2=2
c3=2
For t=-3 To 3 Step .01
    
```





```
u=(-c1*t - c3)/c2
x=x0+t*ed
y=y0 - u*ed
GraphicsWindow.SetPixel(x,y,"")
EndFor
For lmb= -1 To 5 Step .2
t1=m1+lmb*p
t2=m2+lmb*q
t3=1
x1=x0+ed*t1/t3
y1=y0 - ed*t2/t3
GraphicsWindow.FillEllipse(x1 - 3,y1 - 3,6,6)
d1=t2*s3 - t3*s2
d2=t3*s1 - t1*s3
d3=t1*s2 - t2*s1
tt1=d2*c3 - d3*c2
tt2=d3*c1 - d1*c3
tt3=d1*c2 - d2*c1
x2=x0+ed*tt1/tt3
y2=y0 - ed*tt2/tt3
GraphicsWindow.FillEllipse(x2 - 3,y2 - 3,6,6)
GraphicsWindow.DrawLine(xs,ys,x2,y2)
For tm=1 To 50000
EndFor
EndFor
```

Ещё раз вернёмся к вопросам, связанным с систем решением неравенств с несколькими переменными (см. задания 16 и 17), и дадим геометрическое решение задачи линейного программирования для случая двух переменных.

Задание 23. Найти максимум и минимум линейной формы $z = 17x + 23y$ при условии выполнения системы неравенств

$$\begin{aligned}x - y + 1 &\leq 0; \\x + y - 2 &\leq 0; \\-2x + y - 3 &\leq 0.\end{aligned}$$

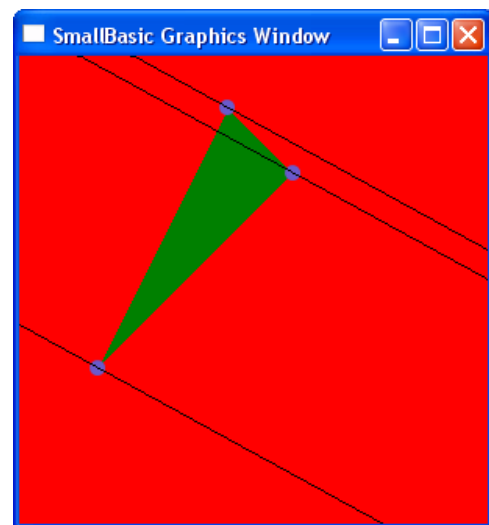
Сначала построим область, заданную системой неравенств. Затем вычислим координаты вершин этой области и проведём через эти вершины линии уровня линейной формы.

```
GraphicsWindow.Width = 300
GraphicsWindow.Height = 300
x0 = 150
y0 = 150
ed = 50
c1 = 17
c2 = 31
a1[1] = 1
a2[1] = -1
a3[1] = 1
a1[2] = 1
a2[2] = 1
a3[2] = -2
```

```

a1[3] = -2
a2[3] = 1
a3[3] = -3
For t = -3 To 3 Step .02
  For u = -3 To 3 Step .02
    xe = x0 + ed*t
    ye = y0 - ed*u
    For i=1 To 3
      znak[i]=a1[i]*t+a2[i]*u+a3[i]
    EndFor
    If znak[1]<0 And znak[2]<0 And znak[3]<0 Then
      GraphicsWindow.SetPixel(xe,ye,"green")
    Else
      GraphicsWindow.SetPixel(xe,ye,"red")
    EndIf
  EndFor
EndFor
For i=1 To 3
  If i<3 Then
    j = i+1
  Else
    j = 1
  EndIf
  d1 = a2[i]*a3[j] - a2[j]*a3[i]
  d2 = a3[i]*a1[j] - a3[j]*a1[i]
  d3 = a1[i]*a2[j] - a1[j]*a2[i]
  tw[i] = d1/d3
  uw[i] = d2/d3
  x[i] = x0 + ed*tw[i]
  y[i] = y0 - ed*uw[i]
  GraphicsWindow.FillEllipse(x[i] - 5,y[i] - 5,10,10)
EndFor
For i=1 To 3
  For t = -3 To 3 Step .02
    u = -c1*t/c2 + c1*tw[i]/c2 + uw[i]
    xe = x0 + ed*t
    ye = y0 - ed*u
    GraphicsWindow.SetPixel(xe,ye,"black")
  EndFor
EndFor

```



А теперь обратимся к вопросам построения прямых, окружностей и кривых второго порядка, а также к нахождению их точек пересечения и проведению касательных.

Задание 24. На плоскости заданы прямая $a \cdot x + b \cdot y + c = 0$ и точка (x_1, y_1) . Построить окружность, касающуюся заданной прямой и имеющей центр в заданной точке.

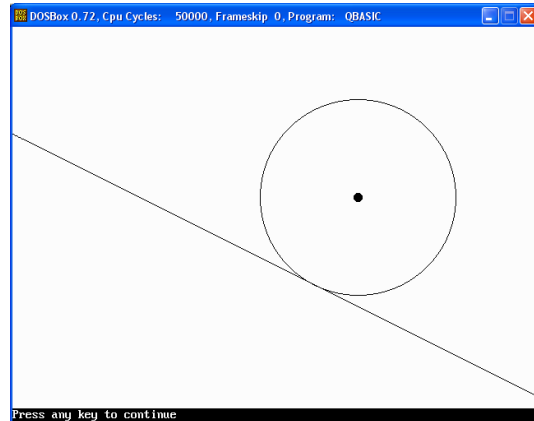
```

x0 = 320
y0 = 240
ed = 100
a = 1

```

```

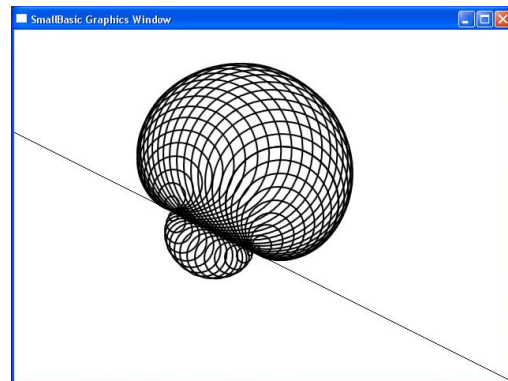
b = 2
c = 1
t1 = 1
u1 = 1 / 3
For t = -3.2 to 3.2 step .01
u = -(a * t + c) / b
x = x0 + ed * t
y = y0 - ed * u
GraphicsWindow.SetPixel(x,y,"")
EndFor
x1 = x0 + ed * t1
y1 = y0 - ed * u1
GraphicsWindow.FillEllipse(x1 - 3,y1 - 3,6,6)
r = ed * Math.Abs((a*t1 + b*u1 + c) / Math.Sqrt(a*a + b*b))
GraphicsWindow.DrawEllipse(x1 - r, y1 - r, 2*r,2*r)
    
```



Задание 25. Построить семейство окружностей, центры которых расположены на заданной окружности, а сами эти окружности касаются некоторой прямой.

```

Pi = 3.14159
n=50
x0 = 320
y0 = 240
ed = 100
a = 1
b = 2
c = 1
For t= -4 To 4 Step .01
u = -(a*t+c)/b
x=x0+t*ed
y=y0 - u*ed
GraphicsWindow.SetPixel(x,y,"")
EndFor
tokr = -.5
uokr = .2
rokr = .7
For i=0 to n
t=tokr+rokr*Math.Cos(2*pi*i/n)
x=x0+t*ed
u=uokr - rokr*Math.Sin(2*pi*i/n)
y=y0 - u*ed
l=(a*t+b*u+c)/ Math.Sqrt(a*a+b*b)
r = Math.Abs(l)*ed
GraphicsWindow.DrawEllipse(x - r,y - r,2*r,2*r)
EndFor
    
```



Задание 26. На плоскости заданы окружность с центром в точке А и радиусом r , а также точка В вне окружности. Провести через эту точку касательную к окружности.

В программе отсутствует ожидаемое решение квадратного уравнения, поскольку используется следующая планиметрическая задача: построить прямоугольный треугольник по гипотенузе и катету.



```
ха=200
ya=250
GraphicsWindow.FillEllipse(ха - 5, ya - 5, 2*5, 2*5)
r=100
GraphicsWindow.DrawEllipse(ха - r, ya - r, 2*r, 2*r)
xb=400
yb=130
GraphicsWindow.FillEllipse(xb - 5, yb - 5, 2*5, 2*5)
Касательные()
Sub Касательные
b=r
xba=ха - xb
yba=ya - yb
c=Math.Sqrt((ха - xb)*(ха - xb)+(ya - yb)*(ya - yb))
a=Math.Sqrt(c*c - b*b)
h=a*b/c
lhb=Math.Sqrt(a*a - h*h)
xh=xb+lhb*xba/c
yh=yb+lhb*yba/c
xc=xh+h*yba/c
yc=yh - h*xba/c
GraphicsWindow.FillEllipse(xc-5, yc-5, 2*5, 2*5)
GraphicsWindow.DrawLine(xb, yb, xc, yc)
xc=xh - h*yba/c
yc=yh+h*xba/c
GraphicsWindow.FillEllipse(xc-5, yc-5, 2*5, 2*5)
GraphicsWindow.DrawLine(xb, yb, xc, yc)
EndSub
```

Задание 27. На плоскости заданы три точки. Провести через них окружность.

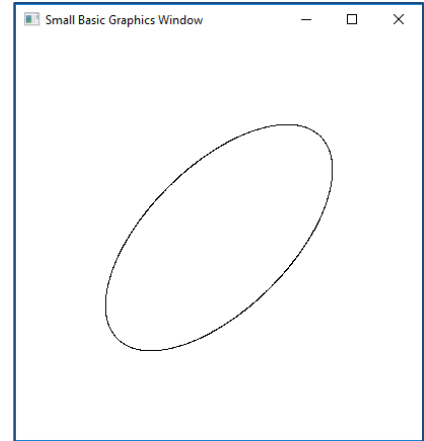
```
x1 = 200
y1 = 100
GraphicsWindow.FillEllipse(x1 - 5, y1 - 5, 10, 10)
x2 = 350
y2 = 200
GraphicsWindow.FillEllipse(x2 - 5, y2 - 5, 10, 10)
x3 = 260
y3 = 300
GraphicsWindow.FillEllipse(x3 - 5, y3 - 5, 10, 10)
ха = (x2 + x3)/2
ya = (y2 + y3)/2
xb = (x1 + x3)/2
yb = (y1 + y3)/2
ka = -(x3 - x2)/(y3 - y2)
kb = -(x3 - x1)/(y3 - y1)
ba = ya - ka*ха
bb = yb - kb*xb
xc = (bb - ba)/(ka - kb)
yc = ka*xc + ba
GraphicsWindow.FillEllipse(xc - 5, yc - 5, 10, 10)
```



```
r = Math.Sqrt((x1 - xc)*(x1 - xc) + (y1 - yc)*(y1 - yc))
GraphicsWindow.DrawEllipse(xc - r, yc - r, 2*r, 2*r)
```

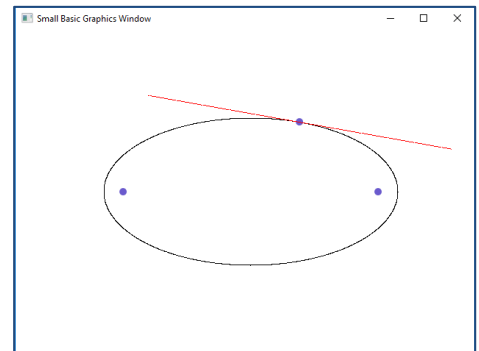
Задание 28. Построить на экране кривую вида $ax^2 + bxy + cy^2 + d = 0$, не закрашивая областей.

```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 400
pi = Math.Pi
x0 = 200
y0 = 200
ed = 100
a = 5
b = -6
c = 5
d = -4
For u = 0 To 2*pi Step pi/1000
  p = Math.Cos(u)
  q = Math.Sin(u)
  lmb = Math.Sqrt(-d/(a*p*p + b*p*q + c*q*q))
  xe = x0 + ed*lmb*p
  ye = y0 - ed*lmb*q
  GraphicsWindow.SetPixel(xe, ye, "")
EndFor
```



Задание 29. Дан эллипс, главные полуоси которого имеют длины a и b . Построить касательную к этому эллипсу в точке M , задаваемой абсциссой x_M и лежащей в верхней полу-плоскости.

```
Pi = Math.Pi
x0 = 320
y0 = 220
a = 200
b = 100
c = Math.Sqrt(a*a - b*b)
xf1 = x0 - c
GraphicsWindow.FillEllipse(xf1 - 5, y0 - 5, 10, 10)
xf2 = x0 + c
GraphicsWindow.FillEllipse(xf2 - 5, y0 - 5, 10, 10)
For ug = 0 To 2*Pi Step pi/1000
  x = x0 + a* Math.Cos(ug)
  y = y0 - b* Math.Sin(ug)
  GraphicsWindow.SetPixel(x, y, "")
EndFor
lmb = 1/3
dx = a*lmb
xm = x0 + dx
ym = y0 - b*Math.Sqrt(a*a - dx*dx)/a
GraphicsWindow.FillEllipse(xm - 5, ym - 5, 10, 10)
p1 = xm - xf1
q1 = ym - y0
s1 = Math.Sqrt(p1*p1 + q1*q1)
```



```

p1 = p1 / s1
q1 = q1 / s1
p2 = xm - xf2
q2 = ym - y0
s2 = Math.SquareRoot(p2*p2 + q2*q2)
p2 = p2 / s2
q2 = q2 / s2
p3 = p1 + p2
q3 = q1 + q2
For n = - 200 To 200
  x = xm + q3*n
  y = ym - p3*n
  GraphicsWindow.SetPixel(x, y, "red")
EndFor

```

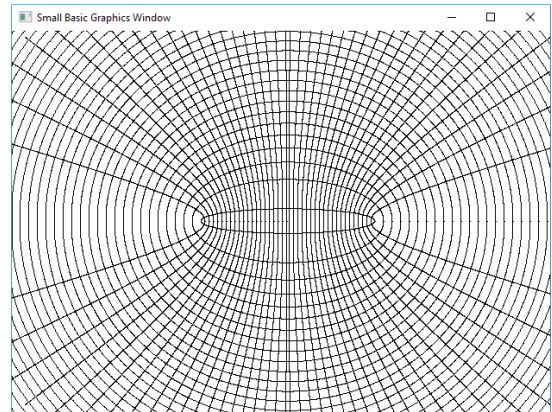
Задание 30. Построить семейства софокусных эллипсов и гипербол.

При построении используется параметризация эллипса с помощью синуса и косинуса, а также параметризация гиперболы гиперболическими функциями.

```

e = 2.71828
x0 = 320
y0 = 220
c = 100
For a = c + 1 To 400 Step 10
  b = Math.SquareRoot(a*a - c*c)
  For u = 0 To 2* Math.Pi Step Math.Pi/1000
    x = x0 + a* Math.Cos(u)
    y = y0 - b* Math.Sin(u)
    GraphicsWindow.SetPixel(x,y, "")
  EndFor
EndFor
For a = 1 To c - 1 Step 5
  b = Math.SquareRoot(c*c - a*a)
  For t = -2 To 2 Step 1/1000
    p = Math.Power(e,t)
    q = 1/p
    x = x0 + a*(p+q)/2
    y = y0 - b*(p - q)/2
    GraphicsWindow.SetPixel(x,y, "")
    x = x0 - a*(p+q)/2
    GraphicsWindow.SetPixel(x,y, "")
  EndFor
EndFor

```



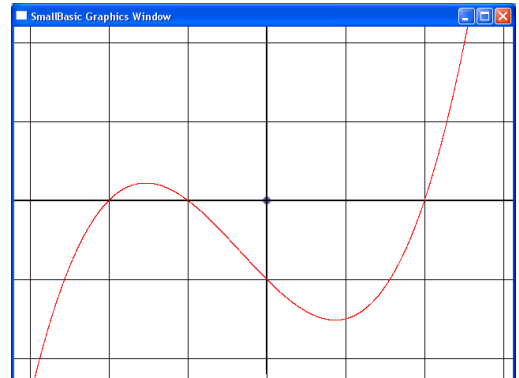
Перейдём теперь к визуализации объектов математического анализа. При этом мы достаточно часто будем пользоваться и методами линейной алгебры. Говоря более конкретно, речь идёт о линейных преобразованиях. Кроме того, мы порой будем рассматривать объекты дифференциальной геометрии и теории дифференциальных уравнений, поскольку соответствующие задачи естественным образом вытекают из задач математического анализа.

Задание 31. Построить на экране график некоторой функции, предварительно разбив экран на единичные квадраты.

```

'Начало координат и единичный отрезок
x0 = 320
y0 = 220
GraphicsWindow.FillEllipse(x0 - 5, y0 - 5, 10, 10)
ed = 100
'Оси системы координат
GraphicsWindow.DrawLine(0, y0, 640, y0)
GraphicsWindow.DrawLine(x0, 0, x0, 440)
'Единичные квадраты
For t = -4 To 4 Step .01
  For u = -3 To 3 Step 1
    x = x0 + t*ed
    y = y0 - u*ed
    GraphicsWindow.SetPixel(x, y, "")
  EndFor
EndFor
For t = -4 To 4 Step 1
  For u = -3 To 3 Step .01
    x = x0 + t*ed
    y = y0 - u*ed
    GraphicsWindow.SetPixel(x, y, "")
  EndFor
EndFor
'Построение графика
For t = -4 To 4 Step .001
  u = (t - 2)*(t + 1)*(t + 2)/4
  x = x0 + t*ed
  y = y0 - u*ed
  GraphicsWindow.SetPixel(x, y, "red")
EndFor

```



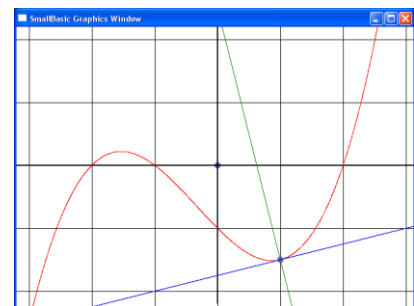
Задание 32. На графике из предыдущего задания выделить точку и провести через неё касательную и нормаль к графику.

Добавим к предыдущей программе следующие строки.

```

tm = 1
um = (tm - 2)*(tm + 1)*(tm + 2)/4
xm = x0 + tm*ed
ym = y0 - um*ed
GraphicsWindow.FillEllipse(xm - 5, ym - 5, 10, 10)
k1 = (3*tm*tm + 2*tm - 4)/4
k2 = -1/k1
b1 = um - k1*tm
b2 = um - k2*tm
For t = -4 To 4 Step .001
  u = k1*t + b1
  x = x0 + t*ed
  y = y0 - u*ed
  GraphicsWindow.SetPixel(x, y, "blue")
  u = k2*t + b2
  x = x0 + t*ed

```



```

y = y0 - u*ed
GraphicsWindow.SetPixel(x, y, "green")
EndFor

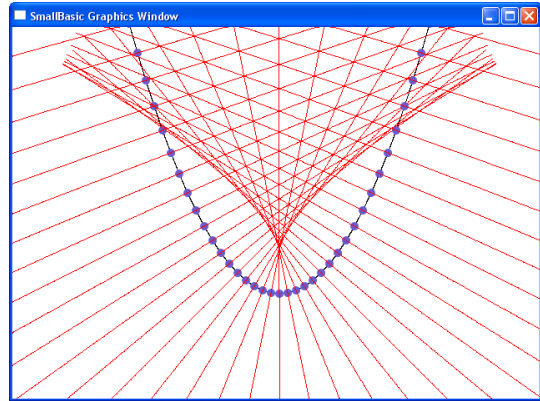
```

Задание 33. Построить семейство нормалей к параболе.

```

x0=320
y0=320
ed=100
For t= -2 To 2 Step .001
  u= t*t
  x=x0+ed*t
  y=y0 - ed*u
  GraphicsWindow.SetPixel(x,y,"")
EndFor
For t1= -2 To 2 Step .1
  u1= t1*t1
  x1=x0+ed*t1
  y1=y0 - e d*u1
  GraphicsWindow.FillEllipse(x1 - 5,y1 -
5,10,10)
  t2=t1+.001
  u2= t2*t2
  k=(u2 - u1)/(t2 - t1)
  p=1/ Math.Sqrt(1+k*k)
  q=k/ Math.Sqrt(1+k*k)
  For t= -400 To 400 Step 1
    x=x1+t*q
    y=y1+t*p
    GraphicsWindow.SetPixel(x,y, "red")
  EndFor
EndFor

```



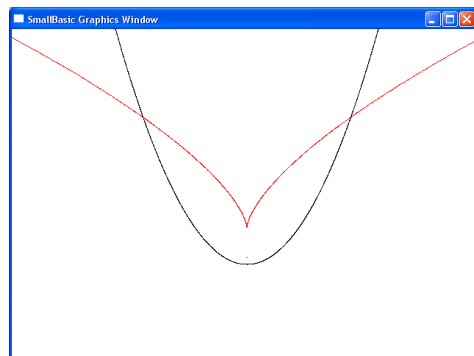
Задание 34. Огибающая нормалей кривой называется каустикой этой кривой. Построить каустик параболы.

Каждая из точек каустики будет приближённо строиться как точка пересечения двух соседних нормалей.

```

x0=320
y0=320
ed=100
For t= -2 To 2 Step .001
  u= t*t
  x=x0+ed*t
  y=y0 - ed*u
  GraphicsWindow.SetPixel(x,y,"")
EndFor
For t1= -4 To 4 Step .001
  u1= t1*t1
  t2=t1+.001
  u2= t2*t2

```





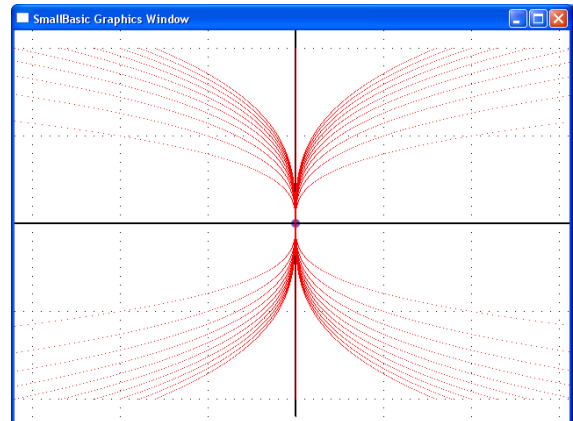
```
k1=(u2 - u1)/(t2 - t1)
t3=t2+.001
u3= t3*t3
k2=(u3 - u2)/(t3 - t2)
If Math.Abs(k1)>.01 Then
a1= -1/k1
a2= -1
a3=u1+t1/k1
Else
a1=1
a2=0
a3= -t1
EndIf
If Math.Abs(k2)>.01 Then
b1= -1/k2
b2= -1
b3=u2+t2/k2
Else
b1=1
b2=0
b3= -t2
EndIf
xk=a2*b3 - a3*b2
yk=a3*b1 - a1*b3
zk=a1*b2 - a2*b1
If Math.Abs(xk)<4*Math.Abs(zk) And Math.Abs(yk)<4*Math.Abs(zk) Then
x=x0+ed*xk/zk
y=y0 - ed*yk/zk
GraphicsWindow.SetPixel(x,y,"red")
EndIf
EndFor
```

Задание 35. По формуле $F(x, y, c) = 0$ построить семейство графиков.

```
'Начало координат и единичный отрезок
x0 = 320
y0 = 220
GraphicsWindow.FillEllipse(x0 - 5, y0 - 5, 10, 10)
ed = 100
'Оси системы координат
GraphicsWindow.DrawLine(0, y0, 640, y0)
GraphicsWindow.DrawLine(x0, 0, x0, 440)
'Единичные квадраты
For t = -4 To 4 Step .1
For u = -3 To 3 Step 1
x = x0 + t*ed
y = y0 - u*ed
GraphicsWindow.SetPixel(x, y, "")
EndFor
EndFor
For t = -4 To 4 Step 1
```

```

For u = -3 To 3 Step .1
  x = x0 + t*ed
  y = y0 - u*ed
  GraphicsWindow.SetPixel(x, y, "")
EndFor
EndFor
'Построение семейства графиков
For u = -2 To 2 Step .01
  For c = 0 To 5 Step .5
    t = u*u*u/c
    x = x0 + t*ed
    y = y0 - u*ed
    GraphicsWindow.SetPixel(x, y, "red")
    y = y0 + u*ed
    GraphicsWindow.SetPixel(x, y, "red")
  EndFor
EndFor
    
```



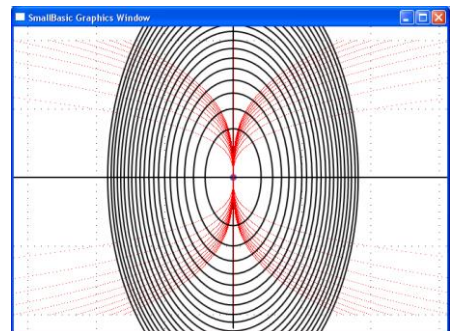
Задание 36. По семейству графиков из предыдущего задания построить ортогональную сеть.

Ортогональная сеть строится для семейства функций, заданных уравнением $y^3 = cx$. Это уравнение следует продифференцировать: $3y^2 y' = c$. Затем исключаем параметр c и получаем дифференциальное уравнение $y = 3y'x$. Далее строится уравнение для ортогональной сети $y \cdot y' = -3x$. Это уравнение с разделяющимися переменными. В итоге ортогональная сеть состоит из эллипсов с полуосями \sqrt{q} и $\sqrt{\frac{q}{3}}$.

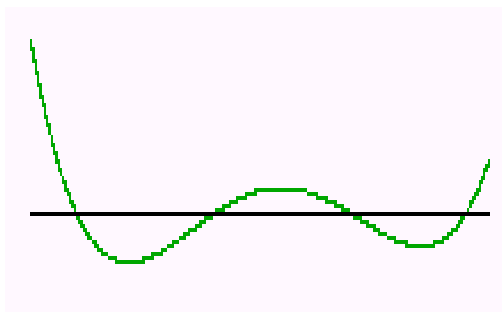
Подробнее см. например [2] .

```

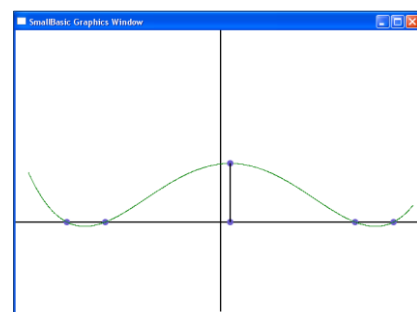
For q = .5 To 3 Step .5
  a = Math.SquareRoot(q/3)
  b = Math.SquareRoot(q)
  For ug = 0 To 2* Math.Pi Step Math.Pi/100
    x = x0 + a* Math.Cos(ug)*ed
    y = y0 - b* Math.Sin(ug)*ed
    GraphicsWindow.SetPixel(x, y, "green")
  EndFor
EndFor
    
```



Задание 37. Построить на отрезке $[-1; 1]$ график алгебраического многочлена по эскизу и найти максимум на этом отрезке.



Эскиз



Результат работы программы

```

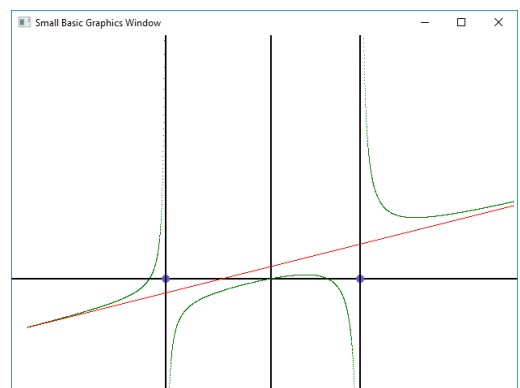
x0 = 320
y0 = 300
ed = 300
GraphicsWindow.DrawLine(x0,0,x0,440)
GraphicsWindow.DrawLine(0,y0,640,y0)
t1 = .9
GraphicsWindow.FillEllipse(x0 + t1*ed - 5, y0 - 5, 10, 10)
t2 = .7
GraphicsWindow.FillEllipse(x0 + t2*ed - 5, y0 - 5, 10, 10)
t3 = -.6
GraphicsWindow.FillEllipse(x0 + t3*ed - 5, y0 - 5, 10, 10)
t4 = -.8
GraphicsWindow.FillEllipse(x0 + t4*ed - 5, y0 - 5, 10, 10)
max = (-1-t1)*(-1-t2)*(-1-t3)*(-1-t4)
xm = -1
For t= -1 To 1 Step .001
x = x0 + ed*t
u = (t-t1)*(t-t2)*(t-t3)*(t-t4)
If u>max Then
    max = u
    xm = x
EndIf
y = y0 - ed*u
GraphicsWindow.SetPixel(x,y,"green")
EndFor
GraphicsWindow.FillEllipse(xm - 5,y0 - 5, 10, 10)
GraphicsWindow.FillEllipse(xm - 5,y0 - max*ed -5, 10, 10)
GraphicsWindow.DrawLine(xm,y0,xm,y0 - max*ed)
    
```

Задание 38. Построить на отрезке $[-3; 3]$ график функции и провести асимптоты.

Уравнение наклонной асимптоты является частным при делении с остатком многочлена, стоящего в числителе, на многочлен, стоящий в знаменателе.

```

x0 = 320
y0 = 300
ed = 100
GraphicsWindow.DrawLine(x0,0,x0,440)
GraphicsWindow.DrawLine(0,y0,640,y0)
t1 = -1.5
t2 = .7
t3 = -1.3
x3 = x0 + t3*ed
GraphicsWindow.FillEllipse(x3 - 5, y0 - 5, 10, 10)
GraphicsWindow.DrawLine(x3,0,x3,440)
t4 = 1.1
x4 = x0 + t4*ed
GraphicsWindow.FillEllipse(x4 - 5, y0 -5, 10, 10)
GraphicsWindow.DrawLine(x4,0,x4,440)
For t=-3.001 To 3 Step .01
x = x0 + ed*t
    
```



```

u = t*(t-1)*(t-2)/(4*(t-3)*(t-4))
y = y0 - ed*u
GraphicsWindow.SetPixel(x,y,"green")
u = t/4 + .15
y = y0 - ed*u
GraphicsWindow.SetPixel(x,y,"red")
EndFor

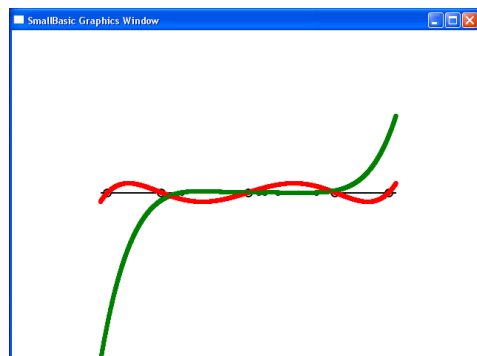
```

Задание 39. Постройте многочлен Чебышева n -й степени (см. [4]), имеющий корни в точках $\cos((2k - 1)\pi/2n)$, где $k = 1, \dots, n$. Постройте также многочлен той же степени со случайными корнями. Оцените уклонения многочленов от оси абсцисс.

```

x0 = 320
y0 = 220
ed = 200
GraphicsWindow.DrawLine(x0 - ed,y0,x0+ed,y0)
pi = Math.Pi
n = 5
For k = 1 To n Step 1
  p[k] = Math.Cos((2*k - 1)*pi/(2*n))
  x = x0 + p[k]*ed
  GraphicsWindow.DrawEllipse(x - 5, y0 - 5, 10, 10)
  q[k] = (Math.GetRandomNumber(200) - 100)/100
  x = x0 + q[k]*ed
  GraphicsWindow.DrawEllipse(x - 3, y0 - 3, 6, 6)
EndFor
For t = -1 To 1 Step .001
  u1 = 1
  u2 = 1
  For k = 1 To n Step 1
    u1 = u1*(t - p[k])
    u2 = u2*(t - q[k])
  EndFor
  x = x0 + t*ed
  y1 = y0 - u1*ed
  y2 = y0 - u2*ed
  GraphicsWindow.BrushColor="red"
  GraphicsWindow.FillEllipse(x - 3,y1 - 3, 6, 6)
  GraphicsWindow.BrushColor="green"
  GraphicsWindow.FillEllipse(x - 3,y2 - 3, 6, 6)
EndFor

```



Задание 40. Отправляясь от доказательства первой теоремы Вейерштрасса, данного С. Н. Бернштейном, построить на отрезке $[0; 1]$ многочлен, аппроксимирующий непрерывную функцию $y = f(x)$ (см. [3], стр. 39).

Многочлен, который мы будем строить, задаётся формулой

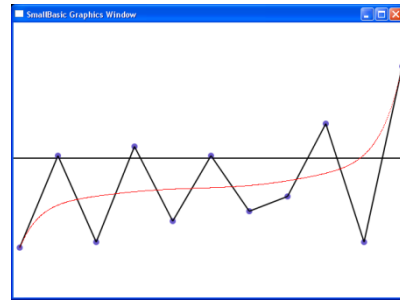
$$B_n(x) = \sum_{k=0}^n C_n^k x^k (1-x)^{n-k} f\left(\frac{k}{n}\right).$$

Выполняя построение многочлена для заданного n , мы на роль функции возьмём логарифмическую с вершинами в точках вида $(\frac{k}{n}; f(\frac{k}{n}))$.

```

x0 = 10
y0 = 220
ed = 620
GraphicsWindow.DrawLine(0,y0,640,y0)
n = 10
For k = 0 To n
  f[k] = (Math.GetRandomNumber(100) - 50)/200
  xf[k] = x0 + ed*k/n
  yf[k] = y0 - ed*f[k]
  GraphicsWindow.FillEllipse(xf[k] - 5,yf[k] - 5,10,10)
EndFor
For k = 1 To n
  GraphicsWindow.DrawLine(xf[k-1],yf[k-1],xf[k],yf[k])
EndFor
For t = 0 To 1 Step .001
  u = 0
  c = 1
  For k = 0 To n
    u = u + c* Math.Power(t,k)* Math.Power(1-t,n-k)*f[k]
    c = c*(n-k)/(k+1)
  EndFor
  x = x0 + t*ed
  y = y0 - u*ed
  GraphicsWindow.SetPixel(x,y,"red")
EndFor

```



Задание 41. Построить график плотности нормального распределения $p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ и график функции распределения $\Phi(x)$.

```

x0 = 320
y0 = 300
ed = 100
e = 2.71828
pi = Math.Pi
dt = .001
f1 = 1/2
f2 = 1/2
GraphicsWindow.DrawLine(x0,0,x0,440)
GraphicsWindow.DrawLine(0,y0,640,y0)
For t = 0 To 3 Step dt
  x1 = x0 + ed*t
  x2 = x0 - ed*t
  u = Math.Power(e, -t*t/2)/ Math.SquareRoot(2*pi)
  y = y0 - ed*u
  f1 = f1 + u*dt
  f2 = f2 - u*dt

```

```

y1 = y0 - ed*f1
y2 = y0 - ed*f2
GraphicsWindow.SetPixel(x1,y,"green")
GraphicsWindow.SetPixel(x2,y,"green")
GraphicsWindow.SetPixel(x1,y1,"red")
GraphicsWindow.SetPixel(x2,y2,"red")
EndFor

```

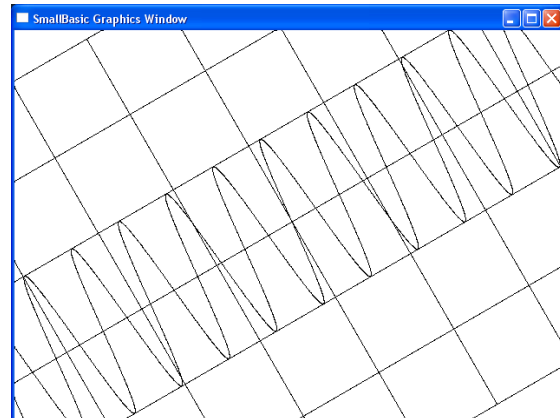
Ряд заданий будет связан с линейными преобразованиями плоскости. В статье [5] дано удобное для использования кинематическое истолкование линейных преобразований. В дальнейшем мы будем ориентироваться именно на него.

Задание 42. Построить график функции в системе координат, наклонённой под заданным углом к горизонтали.

```

x0 = 320
y0 = 220
ed = 100
ug = Math.Pi/6
x1 = x0 + ed* Math.Cos(ug)
y1 = y0 - ed* Math.Sin(ug)
x2 = x0 - ed* Math.Sin(ug)
y2 = y0 - ed* Math.Cos(ug)
For t1 = -4 To 4 Step .01
For t2 = -3 To 3 Step 1
  x = x0 + t1*(x1 - x0) + t2*(x2 - x0)
  y = y0 + t1*(y1 - y0) + t2*(y2 - y0)
  GraphicsWindow.SetPixel(x,y,"")
EndFor
EndFor
For t1 = -4 To 4 Step 1
For t2 = -3 To 3 Step .01
  x = x0 + t1*(x1 - x0) + t2*(x2 - x0)
  y = y0 + t1*(y1 - y0) + t2*(y2 - y0)
  GraphicsWindow.SetPixel(x,y,"")
EndFor
EndFor
For t1 = -4 To 4 Step .001
  t2 = Math.Sin(10*t1)
  x = x0 + t1*(x1 - x0) + t2*(x2 - x0)
  y = y0 + t1*(y1 - y0) + t2*(y2 - y0)
  GraphicsWindow.SetPixel(x,y,"")
EndFor

```



Задание 43. Построить эллипс, наклонённый под заданным углом к горизонтали и имеющий заданные длины полуосей.

```

x0 = 320
y0 = 220
GraphicsWindow.FillEllipse(x0-5,y0-5,10,10)
ed = 100
a = 2

```



```
b = 1
ug = Math.Pi/6
x1 = x0 + ed* Math.Cos(ug)
y1 = y0 - ed* Math.Sin(ug)
GraphicsWindow.FillEllipse(x1-5,y1-5,10,10)
x2 = x0 - ed* Math.Sin(ug)
y2 = y0 - ed* Math.Cos(ug)
GraphicsWindow.FillEllipse(x2-5,y2-5,10,10)
For u = 0 To 2* Math.Pi Step Math.Pi/200
  t1 = a*Math.Cos(u)
  t2 = b*Math.Sin(u)
  x = x0 + t1*(x1 - x0) + t2*(x2 - x0)
  y = y0 + t1*(y1 - y0) + t2*(y2 - y0)
  GraphicsWindow.SetPixel(x,y,"")
EndFor
```

Задание 44. Создать анимационный эффект, связанный с группой вращений плоскости относительно заданной точки.

```
pi=3.14159
nstor=9
m=5
ugol=2*pi*m/nstor
xa=320
ya=240
r=180
For u=0 to 10*pi step pi/20
  Звезда()
  For tmpause=1 to 50000
  EndFor
GraphicsWindow.Clear()
EndFor
Sub Звезда
For n=0 to nstor
x1=xa+r*Math.Cos(ugol*n+u)
y1=ya - r*Math.Sin(ugol*n+u)
x2=xa+r*Math.Cos(ugol*(n+1)+u)
y2=ya - r*Math.Sin(ugol*(n+1)+u)
GraphicsWindow.DrawLine(x1,y1,x2,y2)
EndFor
EndSub
```

Задание 45 Создать анимационный эффект вращения объёмного аксонометрическое изображения усечённой пирамиды.

```
pi=3.14159
n=7
x0=320
y0=300
rniz=150
rwerh=60
h=200
k=1/3
```

```

For u=0 to 10*pi step pi/20
  Пирамида()
  For tmpause=1 to 90000
  EndFor
  GraphicsWindow.Clear()
EndFor
Sub Пирамида
For i=1 To n
  ug=u+i*2*pi/n
  xniz1=x0+rniz* Math.Cos(ug)
  yniz1=y0 - rniz* Math.Sin(ug)*k
  xniz2=x0+rniz* Math.Cos(ug+2*pi/n)
  yniz2=y0 - rniz* Math.Sin(ug+2*pi/n)*k
  xwerh1=x0+rwerh* Math.Cos(ug)
  ywerh1=y0 - rwerh* Math.Sin(ug)*k-h
  xwerh2=x0+rwerh* Math.Cos(ug+2*pi/n)
  ywerh2=y0 - rwerh* Math.Sin(ug+2*pi/n)*k-h
  GraphicsWindow.DrawLine(xniz1,yniz1,xniz2,yniz2)
GraphicsWindow.DrawLine(xwerh1,ywerh1,xwerh2,ywerh2)
  GraphicsWindow.DrawLine(xniz1,yniz1,xwerh1,ywerh1)
EndFor
EndSub

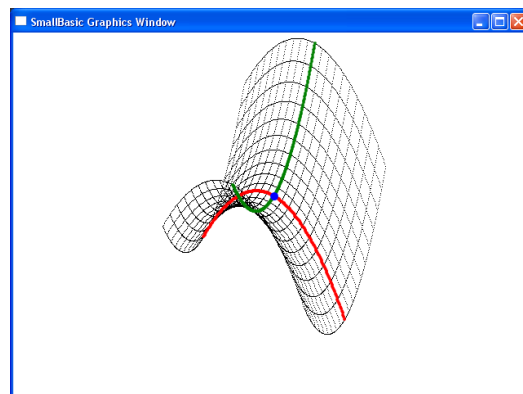
```

Задание 46. Построить поверхность, описываемую уравнением $z = F(x, y)$.
 В нашем случае используется частный вариант уравнения: $z = F_1(x) + F_2(y)$.

```

pi = Math.Pi
u = pi/3
x0 = 320
y0 = 200
ed = 100
k = 1/3
x1 = x0 + ed * Math.Cos(u)
y1 = y0 - ed * Math.Sin(u)
x2 = x0 + ed * Math.Cos(u+pi/2)
y2 = y0 - ed * Math.Sin(u+pi/2)
x3 = x0
y3 = y0 - ed
For t1 = -1 To 1 Step .01
  For t2 = -1 To 1 Step .1
    t3 = t1*t1 - t2*t2
    x = x0 + t1*(x1 - x0) + t2*(x2 - x0) + t3*(x3 - x0)
    y = y0 + t1*(y1 - y0) + t2*(y2 - y0) + t3*(y3 - y0)
    GraphicsWindow.SetPixel(x,y,"")
  EndFor
EndFor
For t1 = -1 To 1 Step .1
  For t2 = -1 To 1 Step .01
    t3 = t1*t1 - t2*t2
    x = x0 + t1*(x1 - x0) + t2*(x2 - x0) + t3*(x3 - x0)
    y = y0 + t1*(y1 - y0) + t2*(y2 - y0) + t3*(y3 - y0)

```



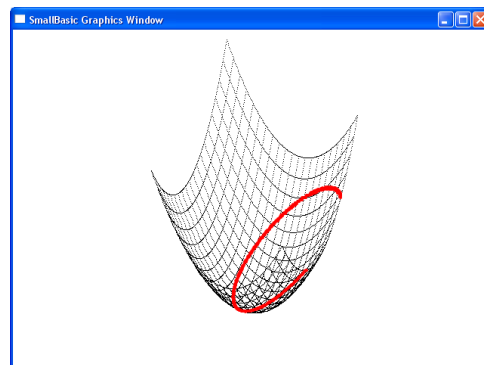
```

GraphicsWindow.SetPixel(x,y,"")
EndFor
EndFor
GraphicsWindow.BrushColor="red"
For t1 = 0 To 0 Step .01
For t2 = -1 To 1 Step .01
    t3 = t1*t1 - t2*t2
    x = x0 + t1*(x1 - x0) + t2*(x2 - x0) + t3*(x3 - x0)
    y = y0 + t1*(y1 - y0) + t2*(y2 - y0) + t3*(y3 - y0)
    GraphicsWindow.FillEllipse(x - 2,y - 2,4,4)
EndFor
EndFor
GraphicsWindow.BrushColor="green"
For t1 = -1 To 1 Step .01
For t2 = 0 To 0 Step .01
    t3 = t1*t1 - t2*t2
    x = x0 + t1*(x1 - x0) + t2*(x2 - x0) + t3*(x3 - x0)
    y = y0 + t1*(y1 - y0) + t2*(y2 - y0) + t3*(y3 - y0)
    GraphicsWindow.FillEllipse(x - 2,y - 2,4,4)
EndFor
EndFor
GraphicsWindow.BrushColor="blue"
t1 = 0
t2 = 0
t3 = t1*t1 - t2*t2
x = x0 + t1*(x1 - x0) + t2*(x2 - x0) + t3*(x3 - x0)
y = y0 + t1*(y1 - y0) + t2*(y2 - y0) + t3*(y3 - y0)
GraphicsWindow.FillEllipse(x - 5,y - 5,10,10)
    
```

Задание 47. Построить сечение некоторой поверхности плоскостью.

```

pi = Math.Pi
u = pi/3
x0 = 320
y0 = 350
ed = 100
k = 1/3
x1 = x0 + ed * Math.Cos(u)
y1 = y0 - ed * Math.Sin(u)
x2 = x0 + ed * Math.Cos(u+pi/2)
y2 = y0 - ed * Math.Sin(u+pi/2)
x3 = x0
y3 = y0 - ed
For t1 = -1 To 1 Step .01
For t2 = -1 To 1 Step .1
    t3 = t1*t1 + t2*t2
    x = x0 + t1*(x1 - x0) + t2*(x2 - x0) + t3*(x3 - x0)
    y = y0 + t1*(y1 - y0) + t2*(y2 - y0) + t3*(y3 - y0)
    GraphicsWindow.SetPixel(x,y,"")
EndFor
EndFor
    
```





```
For t1 = -1 To 1 Step .1
For t2 = -1 To 1 Step .01
t3 = t1*t1 + t2*t2
x = x0 + t1*(x1 - x0) + t2*(x2 - x0) + t3*(x3 - x0)
y = y0 + t1*(y1 - y0) + t2*(y2 - y0) + t3*(y3 - y0)
GraphicsWindow.SetPixel(x,y,"")
EndFor
EndFor
GraphicsWindow.BrushColor="red"
For t1 = -1 To 1 Step .01
For t2 = -1 To 1 Step .01
t3 = t1*t1 + t2*t2
t4 = .2*t1 - .9*t2 +.3
If Math.Abs(t3 - t4) < .01 Then
x = x0 + t1*(x1 - x0) + t2*(x2 - x0) + t3*(x3 - x0)
y = y0 + t1*(y1 - y0) + t2*(y2 - y0) + t3*(y3 - y0)
GraphicsWindow.FillEllipse(x-2,y-2,4,4)
EndIf
EndFor
EndFor
```

Задание 48. Создать анимационный эффект колебания струны.

```
x0 = 320
y0 = 220
ed = 100
pi = Math.Pi
For u = -10*pi To 10*pi Step pi/100
For t = -.5 To .5 Step .005
x = x0 + ed*t
a = Math.Sin(3*pi*t)/10 + Math.Sin(5*pi*t)/2 - Math.Sin(17*pi*t)/5
a = a* Math.Sin(u)
y = y0 - end*a
GraphicsWindow.SetPixel(x,y,"green")
EndFor
For tm = 1 To 50000
EndFor
GraphicsWindow.Clear()
EndFor
```

3. ЗАКЛЮЧЕНИЕ

Представляется уместным в процессе изучения высшей математики использовать задания, целью которых является написание компьютерных программ, визуализирующих изучаемые математические понятия. При этом абстрактное теоретическое знание обретает конкретное образное воплощение. Часто при этом достигается повышение уровня понимания и стимулируется более высокая степень мотивации, направленной на достижение более глубоких математических знаний.

Учащимся могут быть предложены задания различного уровня сложности, от простейших до достаточно сложных. Успешное выполнение любого из таких заданий требует активного использования математических понятий. Сложные задания иногда требуют обра-



щения сразу к нескольким разделам математики, что также поднимает уровень математических знаний.

Кроме того, для современного математика обязательным является владение широким спектром компьютерных технологий, включая и программирование.

ЛИТЕРАТУРА

1. М. Е. Степанов. Некоторые вопросы методики преподавания высшей математики. М., Моделирование и анализ данных. Научный журнал. МГППУ, №1, 2017.
2. А. Г. Школьник. Дифференциальные уравнения. М., Учпедгиз, 1963.
3. Н. И. Ахиезер. Лекции по теории аппроксимации. М. Наука, 1965.
4. П. К. Суетин. Классические ортогональные многочлены. М. Наука, 1976.
5. М. Е. Степанов. Метод сложных движений в компьютерной геометрии. М., Моделирование и анализ данных. МГППУ, №1, 2011.

Работа поступила 14.12.2018г.

ИСПОЛЬЗОВАНИЕ ИГРЫ P LINES ДЛЯ ПСИХОЛОГИЧЕСКОГО ТЕСТИРОВАНИЯ

В.К. Войтов

В статье описывается система психологического тестирования на основе компьютерных игр. Создаются программные средства, отступающие от обычных канонов игры, но позволяющие измерять новые психологические характеристики.

The article describes the system of psychological testing based on computer games. Software tools are created that deviate from the usual canons of the game, but allow you to measure new psychological characteristics.

КЛЮЧЕВЫЕ СЛОВА

Психологическое тестирование, игровая диагностика, компьютерная диагностика, игра PLines.

1. ВВЕДЕНИЕ

В МГППУ на факультете ИТ создаются системы психологического тестирования различной направленности. Разработана система тестирования рабочей памяти человека [1, 2]. Она состоит из десяти тестов и функционирует в Internet. Реализована версия адаптивного теста общего интеллекта Бертлинга-Холлинга [3]. В настоящее время разрабатывается система психологического тестирования на основе компьютерных игр [4, 5]. Этой системе посвящена данная статья.

В качестве основы выбрана игра Lines. Созданный нами вариант игры называется PLines. Тестирование пользователей (игроков) с помощью компьютерных игр упрощает процесс тестирования, так как пользователи часто сами заинтересованы в прохождении теста. Кроме того, в отличие от обычных тестов тестируемые часто выглядят более “раскрепощенными”.

В существующих версиях игры Lines программный код закрыт от пользователя, что не позволяет вносить в код нужные изменения для накопления и последующей обработки статистической информации для психологов. В сети Internet можно найти несколько реализаций этой игры (например, на сайте <http://lines-98.ru>). Разработанная в МГППУ система PLines является полностью самостоятельной программной реализацией игры Lines, что дает возможность вносить в коды программы изменения и настройки нужные для измерения психологических характеристик. Вид и возможности программы соответствуют игре Lines. Это позволяет тестируемым знакомым с классической версией программы легче адаптироваться к системе. При реализации системы использовались языки HTML, JavaScript, PHP, библиотека jQuery, технология Ajax и СУБД MySQL.

На самом деле (согласно потребностям психологов) реализовано три версии системы **PLines**:

- Базовая часть PLines. По функциональным возможностям она включает средства Lines и добавляет средства накопления и обработки данных ориентированные на психологов.
- Версия системы с закономерностями. В ней психологи могут добавлять нужные им закономерности появления шариков.
- Многопользовательская система психологического тестирования.

Первые два варианта уже были использованы для тестирования. Второй и третий варианты в настоящее время дорабатываются.

На рис. 1 изображено окно для входа в систему. Пользователь вводит имя, пароль и группу. Группа является необязательным параметром. Если указанного такого пользователя нет, то он создается и ему присваивается введенный пароль.



Рис. 1. Вход в систему.

2. БАЗОВАЯ ЧАСТЬ PLines

Кратко изложим правила. На поле (обычно из 9 строк и 9 столбцов) появляются различного цвета шарики. Пользователь за один ход может переместить один шарик. Передвигать можно только большие шарики (по горизонтали или вертикали). По диагоналям шарики перемещать нельзя. При перемещении нельзя перепрыгивать через другие шарики. Маленькие шарики являются подсказкой. Они указывают, где и какие большие шарики появятся на следующем шаге. На рис. 4 изображен возможный вид кнопок над игровым полем. В поле “в-игры” задаются возможные варианты подсказок (отсутствие, наличие, монотонный вид). Поддерживаются кнопки “выход”, “результаты” Три шарика справа на рисунке указывают, какие шарики появятся на следующем ходе. При нажатии на кнопку “выход” и окончании игры результат и статистическая информация о прохождении игры заносятся в базу данных, если вы зарегистрировались в системе. Кнопка “результаты” позволяет посмотреть результаты игроков. Для того, чтобы набирать очки, нужно выстроить более четырех шариков одинакового цвета в линию по **диагонали, горизонтали или вертикали**. После назначения очков выстроенные шарики исчезают, а новые шарики не появляются. При каждом ходе без набора очков появляется три новых шарика. Расположение их определяется с помощью датчика случайных чисел. На рис. 2 показан вид игрового поля, а на рис. 3 приведена таблица назначения очков.

Игру можно вызвать по адресу <http://it-span.mgppu.ru/lines/>.

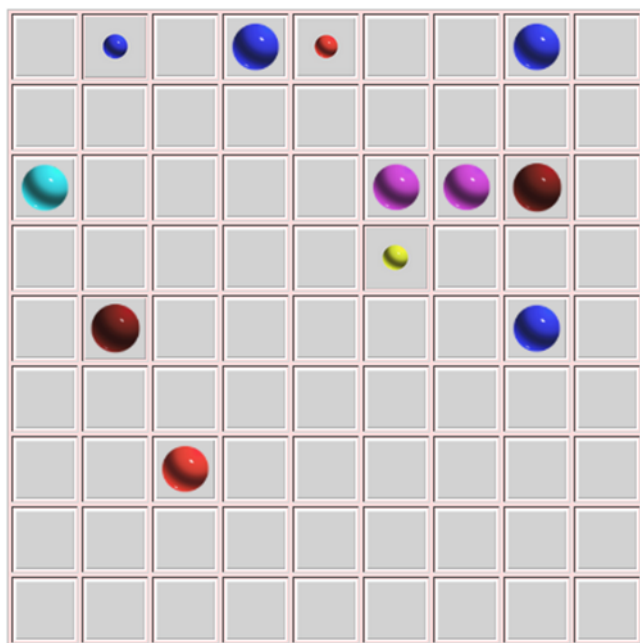


Рис. 2. Вид поля

Число шаров	Очки
5	5
6	12
7	21
8	32
9	45
10	60
11	77
12	96
13	117
14	140
15	165
16	192
17	221

Рис. 3. Назначение очков для сокращенных шариков



Рис. 4. Возможный вид кнопок над игровым полем

В системе поддерживается три вида пользователей: обычный пользователь (игрок, тестируемый), администратор-психолог, системный администратор. **Администратор-психолог** имеет возможность настроить важные параметры игры. К ним относятся, например, такие параметры как ограничения на некоторые временные характеристики, количество используемых цветов для шариков, определение различных подсказок и так далее. Администратор-психолог имеет возможность выбрать интересующие его данные из базы данных для последующей обработки. Например, он может выбрать данные по некоторой группе пользователей для определенного периода времени. Эти данные затем с помощью специальной программы подготавливаются для требуемой статистической обработки. **Администратор-психолог** может также назначить или замаскировать отображение некоторых кнопок или полей (см. рис. 4).

Системный администратор создает базу данных, следит за правильностью занесения и обработки информации. После игры пользователь (игрок) может посмотреть в рейтинговой таблице место, которое он занял.

Система работает в сети Internet. На сервере находятся программные модули системы и база данных, содержащая статистическую информацию о пользователях и взаимодействии их с системой. При вызове пользователем (игроком) программы в локальный компьютер пользователя переносятся модули, поддерживающие работу пользователя и производящие статистическую обработку действий выполняемых пользователем. В конце сеанса статистическая информация о работе пользователя заносится в базу данных на сервере. Такая организация работы позволяет пользователям работать в Internet независимо друг от друга. Последующая обработка накопленной информации производится в автономном режиме администратором системы.

Во время работы пользователя информация в окне игрового поля все время меняется, временная длительность хода для каждого хода разная, количество ходов для каждого сеанса пользователя тоже разное и так далее. Вся эта информация важна для психолога. Для ее хранения используются записи переменной (довольно большой) длины.

На рис. 5 изображена структура таблиц `linesusers` и `linetest`. В таблицу linesusers заносятся важные параметры пользователей: номер пользователя (iduser), его имя (nickname), дата создания (datecreate), групповая принадлежность (gr).

В таблицу linetest заносятся важные параметры прохождения теста: номер пользователя (iduser), номер сеанса (idseance), дата начала прохождения теста (st_date), дата завершения прохождения теста (et_date), число ходов (steptrue), число набранных очков (score), число оставшихся шариков на поле (balls). Поля pmode и nballscolor содержат параметры модификации функционирования системы. Поле strez таблицы linetest рассматривается ниже.

```
CREATE TABLE IF NOT EXISTS `linesusers` (  
  `iduser` int(11) DEFAULT NULL,  
  `nickname` varchar(60) DEFAULT NULL,  
  `datecreate` int(11) DEFAULT NULL,  
  `gr` varchar(12) DEFAULT NULL  
) ENGINE=MyISAM DEFAULT  
CHARSET=cp1251;  
  
CREATE TABLE IF NOT EXISTS `linetest` (  
  `iduser` int(11) DEFAULT NULL,  
  `idseance` int(11) DEFAULT NULL,  
  `st_date` int(11) DEFAULT NULL,  
  `et_date` int(11) DEFAULT NULL,  
  `steptrue` int(11) DEFAULT NULL,  
  `score` int(11) DEFAULT NULL,  
  `balls` int(11) DEFAULT NULL,  
  `rez` int(11) DEFAULT NULL,  
  `strez` blob,  
  `pmode` smallint(6) NOT NULL DEFAULT '1',  
  `nballscolor` smallint(6) NOT NULL DEFAULT '7'  
) ENGINE=MyISAM DEFAULT  
CHARSET=cp1251;
```

Рис. 5. Структура таблиц `linesusers` и `linetest`

2.1. Поле strez

При каждом ходе в некоторую буферную переменную, добавляются следующие параметры (информация о ходе тестируемого):

- счет;
- число сокращенных шариков;
- временная разность начала ходов в 10-х долях секунды. На самом деле время в программе измеряется в миллисекундах, но для уменьшения размера файлов этот параметр учитывается в 10-х долях секунды. Практика показала, что это вполне достаточно;
- число шариков в окне.

В конце сеанса в поле `strez` таблицы `linestest` помещаются данные из буферной переменной тестирования. Поле `strez` имеет тип `blob`, что позволяет помещать в него данные большого размера.

Пример. Вид фрагмента поля `strez` для двух ходов.

:40,40,14,25:52,46,19,19:

Здесь показан фрагмент поля `strez`.

Он содержит параметры 2-х последовательных ходов. Сначала счет и число сокращенных шариков было равно 40. Затем на следующем ходе сократилось 6 шариков. Это дало 12 очков. Время на ход составило 1,9 сек. На экране осталось $25 - 6 = 19$ шариков.

2.2. Обработка результатов тестирования

Для обработки результатов тестирования созданы специальные средства. Администратор-психолог входит под своим именем и паролем в систему и открывается окно “Административная подсистема” (см. рис. 6). Далее он задает параметры для поиска статистических данных в базе. На рис. 6 задан поиск данных тестирования с даты 15-Мая-2016 до даты 26-Мая-2016. Можно также указывать название группы пользователей и диапазон имен пользователей от ник1 до ник2. Для получения данных следует нажать на кнопку “просмотр результатов”.

Полученные данные обрабатываются с помощью специальной программы конвертора, написанной на языке C. Примеры фрагментов представления данных после конвертора для базовой части PLines приведены на рис. 7 а и рис. 7 б. Эти данные уже можно обрабатывать с помощью различных статистических методов.

Здесь в строке 45 “12.15056104.1 счет 172” означает следующее:

15056104 – имя пользователя, 12 – номер записи о тестировании, “.1” указывает сеанс номер 1 для пользователя 15056104, счет 172 – общее количество очков, набранных пользователем. Тот же самый пользователь (строка 49) “12.15056104.2 счет 67” во втором сеансе набрал только 67 очков.

В столбиках приведена информация о ходах (см. выше раздел “Поле `strez`”). Для каждого хода по вертикали вниз расположены: число набранных очков, число сокращенных шариков, время (в 10-х долях секунды), затраченное на ход, число шариков на экране.

Эти данные психологи уже могут подвергать статистическому анализу. Но данная статья преследует только цель описать возможности системы.

Более сложный вариант представления данных смотрите в разделе “Обработка закономерностей”.

2.3. Дополнительные возможности Администратора-психолога

Администратор-психолог имеет возможность назначить параметры функционирования.

Можно задать один из возможных четырех режимов работы:

0 – не отображаются маленькие шарики; 1 – обычный вид; 2 – все маленькие шарики серые; 3 – маленькие шарики отображаются только в правом верхнем углу и не отображаются в основном окне.

Администратор-психолог может назначить число возможных цветов шариков (от 1 до 7).

Кроме того, имеется возможность указать о необходимости накопления статистики об обращении тестируемых к таблице рейтингов.

2.4. Таблица результатов

На рис. 8 отображено начало Таблицы результатов. Каждый пользователь системы может посмотреть, нажав кнопку “результаты” (см. рис. 4), место, которое он занимает в этой таблице. Такая возможность позволяет повысить интерес пользователя к повторному прохождению теста, а психологу это дает (вместе со статистикой обращения к этой таблице) новую информацию для обработки.

2.4. Административная подсистема

просмотр результатов

группа: d1: 15-Май-2016 d2: 27-Май-2016

ник1: ник2:

просмотр пользователей

возврат на стартовую страницу

Рис. 6. Вид окна Административной подсистемы.

41	11.15056102.11 счет 40	0	0	0	0	0	0	0	0	0	0	5	5	10	10	10	15	15	15	20	20	20	20	20	25	30	30
42	шары 40	0	0	0	0	0	0	0	0	0	0	5	5	10	10	10	15	15	15	20	20	20	20	20	25	30	30
43	время 155	387	16	25	41	14	36	43	16	17	13	11	21	19	41	21	37	220	29	14	50	75	30	19	22	27	39
44	все шары	6	9	12	15	18	21	24	27	30	33	28	31	26	29	32	27	30	33	28	31	34	37	40	35	30	33
45	12.15056104.1 счет 172	0	0	0	0	0	5	5	10	10	15	15	15	15	20	20	20	20	20	20	20	25	30	30	35	35	35
46	шары 122	0	0	0	0	0	5	5	10	10	15	15	15	15	20	20	20	20	20	20	20	25	30	30	35	35	35
47	время 337	142	44	31	26	32	23	39	27	23	20	24	31	23	26	76	63	24	33	18	21	18	26	67	38	46	53
48	все шары	6	9	12	15	18	13	16	11	14	9	12	15	18	13	16	19	22	25	28	31	26	24	27	22	25	28
49	13.15056104.2 счет 67	0	0	0	0	0	0	0	0	0	5	5	5	5	5	5	10	15	20	20	20	20	25	25	25	25	25
50	шары 43	0	0	0	0	0	0	0	0	0	5	5	5	5	5	5	10	15	20	20	20	20	25	25	25	25	25
51	время 152	34	21	18	17	51	23	24	18	24	81	77	14	27	16	256	11	49	37	26	16	14	15	47	44	45	18
52	все шары	6	9	12	15	18	21	24	27	30	25	28	31	34	37	40	35	30	25	28	31	34	29	32	35	38	41
53	14.15056104.3 счет 79	0	0	0	0	0	0	0	0	0	21	21	21	21	21	42	47	47	47	52	52	52	57	57	57	57	57
54	шары 45	0	0	0	0	0	0	0	0	0	7	7	7	7	7	14	19	19	19	24	24	24	29	29	29	29	29
55	время 178	74	22	19	17	23	18	25	24	25	34	47	18	24	51	49	70	48	17	13	126	23	36	54	18	22	28
56	все шары	6	9	12	15	18	21	24	27	30	23	26	29	32	35	28	23	26	29	24	27	30	25	28	31	34	37

Рис. 7 а. Фрагмент полученных данных для базовой части системы.

73.15056413.3 счет 95	0	0	20	25	25	25	25	25	30
шары 95	0	0	20	25	25	25	25	25	30
время 433	50	31	39	27	58	64	27	20	51
все шары	6	9	34	29	32	35	38	41	36
74.15056413.4 счет 85	0	0	25	30	30	30	35	35	35
шары 85	0	0	25	30	30	30	35	35	35
время 466	402	168	19	18	32	19	19	122	24
все шары	6	9	26	21	24	27	22	25	28

Рис. 7 б. Другой фрагмент полученных данных для базовой части системы.

***** Результаты *****

№	пользователь	score	balls	date	time	gr
1	krisssss	1001	879	19-Dec-2015 17:21:16	3797	2
2	krisss	971	871	20-Dec-2015 01:21:55	2821	2
3	krisssss	919	785	23-Dec-2015 01:51:53	3103	2
4	Ермаков Сергей	681	589	19-Dec-2015 00:14:24	2228	МГППУ
5	15056207	559	391	16-Dec-2015 21:36:58	1625	62
6	krisss	548	408	19-Dec-2015	2566	2

Рис. 8. Начало таблицы результатов.

3. ВАРИАНТ С ОБРАБОТКОЙ ЗАКОНОМЕРНОСТЕЙ

3.1. Закономерности

Вторая версия системы позволяет после каждого хода без набора очков генерировать три новых шарика не случайным образом, а согласно некоторым закономерностям (составленным по некоторому правилу). Если тестируемый понимает правила, которые лежат в основе функционирования закономерности, это помогает ему набирать больше очков. Кроме того, создано специальное окно, в которое тестируемый должен записать информацию о найденных им закономерностях. Накопленная в базе данных информация помогает оценить способности человека. Разработано несколько закономерностей. Они пронумерованы. При проведении тестирования администратор-психолог выбирает нужные (нужный набор номеров) закономерности, назначает для них время действия. В качестве первой закономерности можно также для тренировки назначить закономерность с номером ноль. Это означает отсутствие закономерностей.

Было проведено тестирование школьников 7-х классов.

На рис. 9 изображено окно, в котором Администратор-психолог назначает параметры функционирования. Параметр режим такой же, как для базовой части (см. выше раздел “Дополнительные возможности Администратора-психолога”).

В поле “посл” задается последовательность номеров закономерностей, которые будут предъявляться. В поле “вр. законом” задается время, отводимое на одну закономерность. В поле “общее время законом” задается общее время для сеанса. Время задается в миллисекундах. Поэтому в таблице на одну закономерность отведено 4 минуты, а на весь сеанс - 40 минут. В поле “рейтинг” указывается, следует ли сохранять информацию об обращении к таблице результатов (1 – да, 0 – нет). В поле “число цветов” указывается число генерируемых цветов. В поле “порядок цветов” указываются (если нужно) нужные цвета. Порядок цветов: 0 – нет порядка, о – коричневый, з – зеленый, к – красный, а – цвет морской волны, ж – желтый, с – синий, ф – фиолетовый.

На рис. 10 приведен пример логики генерации одной из закономерностей. Рассмотрим правила. Закономерность представляет собой генерацию прямоугольных треугольников с

коричневым шариком в прямом угле. Треугольники вращаются по часовой стрелке. Здесь маленьких шариков нет. Положение треугольников определяется с помощью датчика случайных чисел. В качестве ходов использовалось передвижение шариков в нижних двух строках.

При работе с закономерностями над окном с шариками создается окно для занесения обнаруженных особенностей данной закономерности (см. рис. 11). Если тестируемый обнаружил какие-то особенности генерации шариков, он пишет об этом в окно для занесения ответа и нажимает кнопку “Отправить ответ”. После этого система переходит к следующей закономерности.

Если истекло время, отведенное для закономерности, а испытуемый не отправил ответ, возникает окно, приведенное на рис. 12. Таким образом, для каждой закономерности тестируемый может сообщить, какие особенности данной закономерности он обнаружил. Информация о ходах для каждой закономерности имеет такой же вид как для “базовой части” и представляет собой запись переменной длины. Аналогично, заносимая тестируемыми информация о закономерностях, оформляется в виде записей переменной длины. Все эти данные собираются вместе и помещаются в поле *strez* (см. выше), которое имеет более сложную структуру, чем для “базовой части”.

Просмотр таблицы параметров					Модификация таблицы параметров						
№	группа	режим	число цветов	рейтинг	№ закон.	порядок цветов	начало зак	посл	вр законом.	общее время	
1	все	1	7	1	4	озкажсф	0	3, 7, 4, 2, 6, 12, 50	240000	2400000	

Рис. 9. Окно, в котором Администратор-психолог назначает параметры функционирования

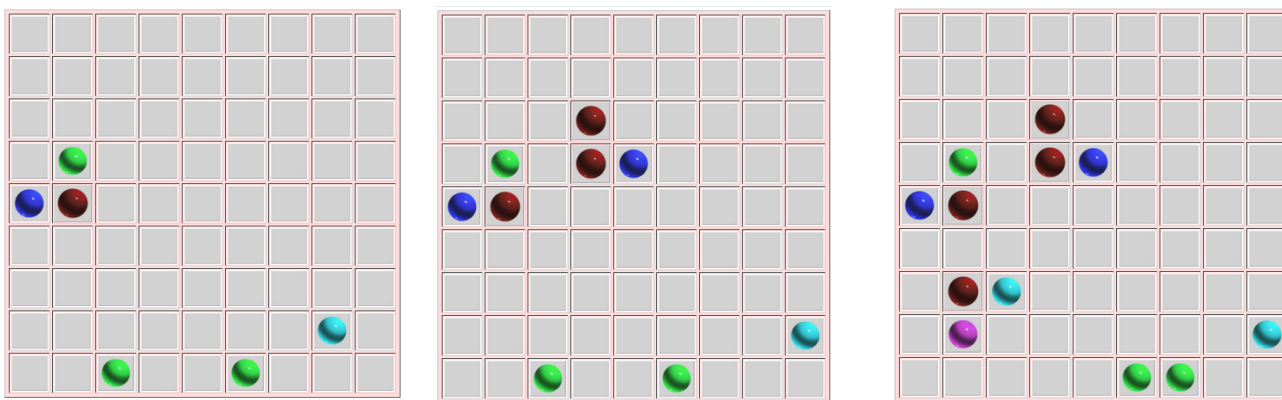


Рис. 10. Пример логики генерации одной закономерности.

В качестве ходов использовалось передвижение шариков в нижних двух строках. Закономерность представляет собой генерацию прямоугольных треугольников с коричневым шариком в прямом угле. Треугольники вращаются по часовой стрелке. Здесь маленьких шариков нет.

	Отправить ответ_
--	------------------

Рис. 11. Окно для занесения обнаруженных особенностей данной закономерности.

Истекло время работы с правилом

Постарайся написать, как тебе кажется, в чем могло состоять правило и нажми кнопку "Отправить ответ" .
Только после того, как ты напишешь ответ и нажмешь кнопку "Отправить ответ", произойдет переход к игре по новому правилу.

Отправить ответ:

Рис. 12. Окно, возникающее, если истекло время, отведенное для закономерности, а испытуемый не отправил ответ

Структура хранения данных в случае с закономерностями в поле `strez` значительно более сложная чем для базовой части поэтому опишем основные хранимые данные базы данных в форме Бэкуса — Наура.

Каждый тестируемый может совершить несколько сеансов.

Фигурные скобки обозначают (`{ ... }`), что «обрамленная» ими конструкция может повториться (но не обязательно) некоторое количество раз. Квадратные скобки обозначают (`[...]`), что «обрамленная» ими конструкция может отсутствовать.

```

<Информация-о-работе-тестируемого>:= [Тренировочная-игра]
    <Информация-о-сеансе-тестируемого> {<Информация-о-сеансе-тестируемого> }
<Информация-о-сеансе-тестируемого>:=
    <Параметры-сеанса> <Сообщения-тестируемого> <Информация-о-ходах-
тестируемого>
<Параметры-сеанса>:= <Ник-тестируемого> <Дата-сеанса><Номер-сеанса>
    <Время-начала-работы-с-первой-закономерностью>
<Сообщения-тестируемого>:= <Сообщение-тестируемого> {<Сообщение-тестируемого> }
<Сообщение-тестируемого>:= <Идентификатор-закономерности>
    <Время-завершения-работы-с-закономерностью> <Текст-сообщения> .
<Информация-о-ходах-тестируемого>:= <Информация-о-ходе> {<Информация-о-ходе> }
<Информация-о-ходе>:= <Счет> <Число сокращенных шариков>
    <Время-на-ход-в-10-х-долях-секунды> <Число-оставшихся-шариков-в-окне>
  
```

Полученные данные помещаются в текстовый файл и обрабатываются с помощью специальной программы “конвертор”, подобной той, о которой говорилось выше. Составляется таблица, упорядоченная по сеансам пользователей. На рис. 13 приведен фрагмент таблицы для обработки данных с закономерностями.

Во время работы с закономерностями тестируемые играют в PLines и набирают очки. Время в первой колонке задано в секундах (начало, конец и длительность сеанса). Имя тестируемого “15056106_1”, сеанс номер 2, время сеанса задано в секундах 2502 (оно включает и время ответа тестируемых).

Время в строках "время:" задано в десятых долях секунды (по шагам). Данные в столбцах идут по шагам: число набранных очков, число сокращенных шариков, время (в 10-х долях секунды), затраченное на ход, число шариков на экране. Буква `z` означает начало новой закономерности. Были назначены и использованы закономерности `z0`, `z51`, `z22`, `z3`, `z7`, ...

Для тренировочной игры, которая предлагается в самом начале, используется специальное обозначение `z0`. Тренировочная игра может отсутствовать.

Для ответов тестируемых (кнопка “Отправить ответ”) о выявленных закономерностях составляется другая таблица. Она также упорядочена по сеансам пользователей.

7.ник=15056106_1.сеанс:2 время:2502	z0 счет: 15	0	0	0	0	0	0	10	10	10	10	10	10	15	15	15
нач:1480578782 кон:1480581284	шары: 15	0	0	0	0	0	0	10	10	10	10	10	10	15	15	15
	время: 1837	23	26	16	46	47	16	155	251	22	17	10	11	33	103	25
	все шары	6	9	12	15	18	21	53	56	59	62	65	68	63	66	69
	z51 счет: 124	0	0	0	0	5	5	25	30	30	30	30	30	35	35	35
	шары: 112	0	0	0	0	5	5	25	30	30	30	30	30	35	35	35
	время: 2996	662	29	15	34	23	39	13	12	56	9	10	10	12	67	15
	все шары	6	9	12	15	10	13	29	24	27	30	33	36	31	34	37
	z22 счет: 116	0	0	0	0	0	0	25	25	25	25	30	30	30	30	30
	шары: 102	0	0	0	0	0	0	25	25	25	25	30	30	30	30	30
	время: 2739	385	37	23	19	65	30	49	20	83	116	18	120	47	23	12
	все шары	6	9	11	14	17	20	26	29	32	35	32	35	36	38	40
	z3 счет: 65	0	0	0	5	5	5	20	25	25	25	30	30	30	30	30
	шары: 65	0	0	0	5	5	5	20	25	25	25	30	30	30	30	30
	время: 1963	172	16	42	19	37	37	13	20	73	27	18	49	27	13	22
	все шары	6	9	12	7	10	13	37	32	35	38	33	36	39	42	45
	z7 счет: 110	0	0	0	0	0	0	25	25	25	30	35	35	35	35	35

Рис. 13. Фрагмент таблицы для обработки данных с закономерностями.

4. МНОГОПОЛЬЗОВАТЕЛЬСКИЙ ВАРИАНТ

В рассмотренных выше версиях одновременно могло работать много пользователей, но каждый из них работал со своей реализацией программы. В многопользовательском варианте два игрока играют в одну игру (как два шахматиста играют одну партию).

При многопользовательском режиме поддерживаются два экрана. Пользователи играют на разных компьютерах и осуществляют ход по очереди.

Разработанная новая версия программы в отличие от предыдущей версии разбивает пользователей на пары. Пользователи пары совместно работают над одной реализацией игры. Для них на экране компьютера отображаются данные их совместной игры

4.1. Передача данных между игроками

На рис. 14 изображены экраны пользователей пары, играющей в многопользовательском режиме. Соединение установлено между пользователями “qq” и “qq2”. Право хода у пользователя “qq2”. После того как пользователь с правом хода сделает ход, у него появляется надпись “НЕТ ХОДА!”, а у его партнера появляется надпись “НАШ ХОД!”. При этом содержимое окна от сделавшего ход игрока переносится партнеру. В отличие от рассмотренных выше вариантов (I и II) информация о виде общего окна хранится на сервере, а не в локальных компьютерах. При этом используются библиотека jQuery [6], технология Ajax [7]. В технологии Ajax используется “обратный вызов”, что позволяет избежать дрожания в верхней строке экрана, которое возникает, если не использовать “обратный вызов”.

Так же как в предыдущей версии программы статистика работы пользователей с программой (игрой) заносится в базу данных. В Системе поддерживается три вида пользователей: обычный пользователь (игрок), администратор-психолог, системный администратор. Администратор-психолог имеет возможность настроить некоторые важные параметры игры. К ним относятся, например, такие параметры как задание ограничений на некоторые временные характеристики, количество используемых цветов, определение различных подсказок и так далее. Созданы средства обработки данных базы и предоставления их психологам в отформатированном виде.

Пользователи пары получают возможность хода по очереди. После произведения хода пользователем вся информация (конфигурация) о состоянии системы передается другому пользователю (по паре). Ему же передается право хода, а возможность хода прежним пользователем блокируется.

4.2. Окно для отправки коротких сообщений

Для каждого пользователя поддерживается окно для отправки коротких сообщений партнеру по паре. Это средство помогает им совместно достигать лучших результатов. Посланные сообщения заносятся в базу данных. Психолог может проанализировать работу каждого пользователя в отдельности. Анализ передаваемых сообщений дает возможность анализа совместной работы в паре.

На рис. 15 а и рис. 15 б изображены соответственно окно сообщений пользователя “qq” и пользователя “qq2”. Для инициирования окон каждый пользователь должен у себя нажать на кнопку “запуск сообщений”. Каждое сообщение отправляется при нажатии на кнопку “отправить сообщение”. Внизу под кнопкой располагается окно, в которое надо помещать сообщение. Ниже расположена “Лента сообщений”. Это список отправленных и принятых сообщений.

В поле “период цикла” задается период времени в миллисекундах, через которое будет обновляться окно сообщений.

4.3. Установление связи пользователей

В начале работы пользователь должен указать, что он находится в сети Internet и выбрать себе пару. В целях упрощения тестирования можно использовать фиксированный набор пар.

После входа в систему при нажатии на кнопку “выбор контакта” появляется окно, изображенное на рис. 16. В нем отображается список пользователей для соединения. Если нужный пользователь активен, то при щелчке мышью по его имени устанавливается соединение и отображается окно с шариками (см. рис. 17).

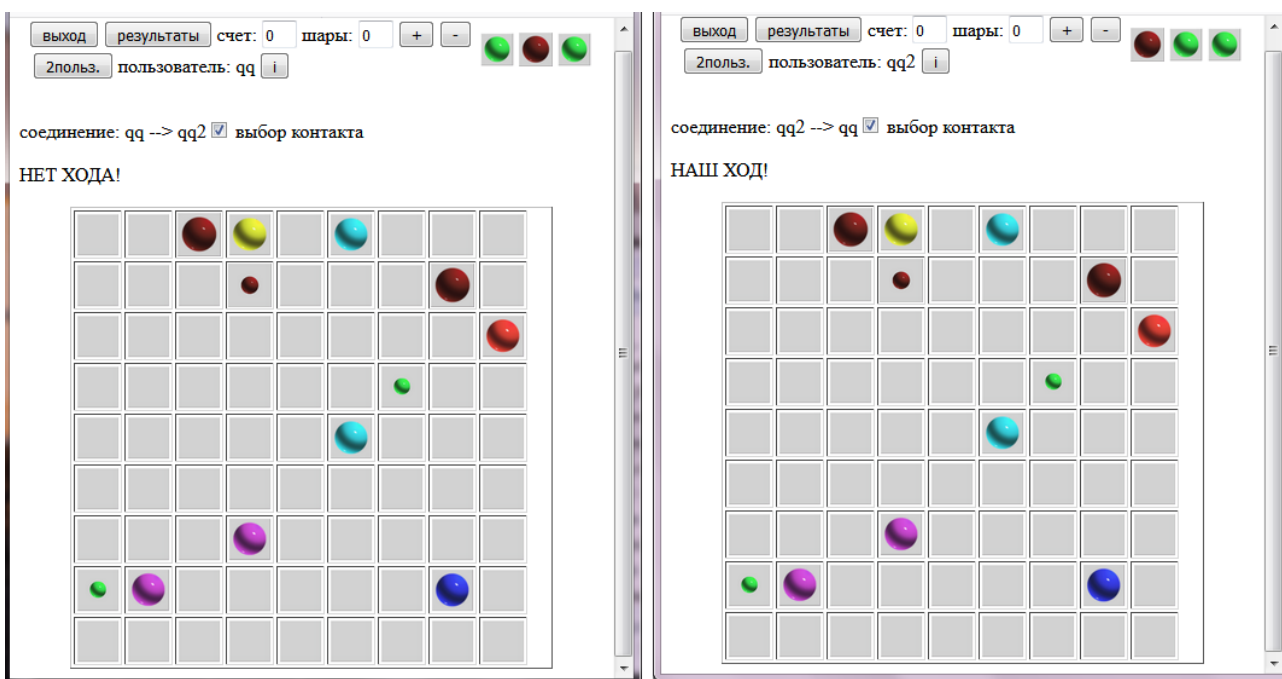


Рис. 14. Вид экранов пользователей пары, играющей при многопользовательском режиме. Право хода у пользователя “qq2”.

запуск сообщений период цикла: 1000 врм: 744:20:57:45

пользователь для получения наших сообщений (user): qq2

отправить сообщение 0 0 сообщение для отправки:
Не будем спешить.

Лента сообщений

```
<отправ> 64: 1 Привет от qq!
<получено> uz 2 : 1 : Добрый день! Я qq2.

<отправ> 64: 2 Не будем спешить.
<получено> uz 2 : 2 : Согласен.
```

Рис. 15 а. Окно сообщений пользователя qq.

запуск сообщений период цикла: 1000 врм: 915:21:0:34

пользователь для получения наших сообщений (user): qq

отправить сообщение 0 0 сообщение для отправки:
Согласен.

Лента сообщений

```
<получено> uz 64 : 1 : Привет от qq!

<отправ> 2: 1 Добрый день! Я qq2.
<получено> uz 64 : 2 : Не будем спешить.

<отправ> 2: 2 Согласен.
```

Рис. 15 б. Окно сообщений пользователя qq2.

2польз. пользователь: qq

выбор контакта

Пользователи:	Активные:
qq	1
qq2	1
qq3	1
qq4	1
t1	
t2	1
t3	

Рис. 16.

Список пользователей для соединения.

2польз. пользователь: qq i

соединение: qq --> qq2 выбор контакта



Рис. 17.

Вид после соединения с польз. "qq2".

5. ЗАКЛЮЧЕНИЕ

Система развивается в сторону создания средств более интересующих психологов. При этом происходит создание программных средств, отступающих от обычных канонов игры, но позволяющих измерять новые психологические параметры. Уже проводилось тестирование в базовом варианте системы и в режиме с обработкой закономерностей. Тестирование в многопользовательском варианте еще не производилось. Полученные результаты анализируются психологами.



ЛИТЕРАТУРА

1. Войтов В.К., Косихин В.В., Ушаков Д.В. Рабочая память как перспективный конструкт когнитивной психологии и методы его измерения // Моделирование и анализ данных. 2015. № 1. С. 57-78.
2. Войтов В.К. Расчет значений сложностей заданий для адаптивного теста / В.К. Войтов // Экспериментальная психология. 2013. Т.6, № 2. С. 120-128.
3. Войтов В. К. Особенности программной реализации адаптивного теста общего интеллекта Берглинга-Холлинга. - Моделирование и анализ данных, 2013, № 1, с. 110-115.
4. Войтов В.К. Многопользовательские системы психологического тестирования на основе компьютерных игр. В книге: Нейрокомпьютеры и их применение XVI Всероссийская научная конференция : тезисы докладов. 2018. С. 225-226.
5. Войтов В.К. Программное обеспечение для диагностических компьютерных игр. В книге: Нейрокомпьютеры и их применение Тезисы докладов. 2017. С. 122-В.
6. Хольцнер С. jQuery. Практическое применение. / Стивен Хольцнер. М.: Эксмо, 2010. – 334 с.
7. Эспозито Д. Разработка веб-приложений с использованием ASP.net и Ajax. СПб.: Питер. 2012. – 400 с.

Работа поступила 02.09.2018г.

ПРЕОБРАЗОВАТЕЛЬ ДИСКРЕТНОЙ ИНФОРМАЦИИ И ПУСТОЕ СЛОВО

ЙОЗЕФ БОКР

В статье рассматривается преобразование слов конечной длины конечным автоматом, уделяется особое внимание преобразованию пустых слов.

The article deals with the transformation of words of finite length by finite automata, special attention is paid to the transformation of empty words.

КЛЮЧЕВЫЕ СЛОВА

Преобразователь, слово в алфавите, пустое слово, конечный автомат, возмущение.

1. ВВЕДЕНИЕ

В статье рассматривается преобразование слов конечной длины конечным автоматом, уделяя особое влияние преобразованию пустых слов.

Как осуществляется восприятие пустого, ненаблюдаемого слова преобразователем и как может преобразователь выдавать пустое слово? Что собой, по сути дела, представляет пустое слово? Можно считать возможным обозначение непосредственно ненаблюдаемых входных слов или неизмеримых возмущений, воздействующих на преобразователь, через символ, зарезервированный для пустого слова? На эти, часто задаваемые вопросы, попытается предлагаемая статья ответить.

2. ПОНЯТИЕ

Определение 1: Как известно, понятие – это форма мышления, отражающая группу однородных предметов в их существенных признаках. Содержанием понятия называется совокупность существенных признаков предметов, которая мыслится в данном понятии. Совокупность предметов, которые мыслятся в понятии, образует объем понятия [1].

Пустое понятие – это понятие, объем которого соответственно содержит один предмет, пустой. Поскольку имеет силу закон обратного отношения между объемом и содержанием понятия, содержание пустого понятия располагает всеми возможными признаками несуществующего предмета, в их числе и противоречивыми признаками, например, отрицательное натуральное число. Заметим, что т.н. «круглый квадрат» не является пустым понятием.

Определение 2: «Ничто» (с латинского nihil) – это единичное понятие, выражающее отсутствие какого-либо конкретного предмета, т.е. его содержание - это свойство быть недостающим конкретным предметом и объем «ничего» содержит то, что отсутствует, т.е. в

объем «ничего» входит пустое множество. Единичное понятие «манипулируемо», т.е. может входить в некоторые операции над понятиями, в отличие от пустого понятия.

«Ничто» не относится к предметам действительности, а является лишь «человеческим» понятием, не обозначающем пустоту, ибо никакое «ничто» не существует.

Понятие выражается в словах и в словосочетаниях.

3. ПРЕОБРАЗОВАТЕЛЬ ДИСКРЕТНОЙ ИНФОРМАЦИИ

Дискретную информацию, носителем которой является дискретный сигнал, можно считать алфавитной, обозначая значения отдельных уровней (квант) сигнала буквами некоторого алфавита. Число квант дискретного сигнала конечно.

Рассмотрим логический стационарный динамический **преобразователь** дискретной информации [2,3]. Основным свойством логического динамического преобразователя является наличие дискретного конечного множества состояний и скачкообразного (мгновенного) перехода преобразователя из одного состояния в другое. Разумеется, для любого реального преобразователя длится действительный переход между состояниями ненулевое, конечное время $\Delta\tau$, ибо преобразователь в течение перехода между состояниями находится в ненаблюдаемом промежуточном состоянии. Отождествив промежуточное состояние либо с исходным, либо с конечным состоянием перехода и учитывая, что переходы между соответственно промежуточным и конечным состояниями и также исходным и промежуточным состояниями мгновенные, добьемся того, что, с одной стороны, реакция преобразователя длится весь переход, и с другой, что промежуточные состояния в таблице переходов служат показателями на достижимое «устойчивое» состояние.

Характерной особенностью логических преобразователей является «дискретность» времени его работы. Дискретные моменты ν времени отождествляются с натуральными числами, считая нулевой момент времени начальным. Выделяются некоторые моменты $\nu, \nu+1, \nu+2, \dots$ реального времени, в которые рассматриваются состояния преобразователя и предполагается, что последовательность состояний и внешних воздействий описывает действие преобразователя. Выбор моментов времени представляет собой последовательность либо с постоянным, либо с переменным шагом $\tau (t_i = t_{i-1} + \tau)$ в зависимости от того, определяются ли моменты времени тактовым генератором ($\tau = \text{const} \gg \Delta\tau$) или выходом автосинхронного преобразователя ($\tau = \Delta\tau$), либо моменты соответствуют «устойчивым» состояниям, и тогда интервалы между последовательными моментами времени различны ($\tau = \Delta\tau$).

Ограничимся детерминированными преобразователями информации.

4. СЛОВО В АЛФАВИТЕ

Определение 3: Пусть A некоторый алфавит. Конечная последовательность $\alpha: \{0, 1, \dots, l-1\} \rightarrow A: 0 \mapsto a_{i_1}, 1 \mapsto a_{i_2}, \dots, l-1 \mapsto a_{i_l}$ называется словом [3,4,5] конечной длины ($|\alpha| = l$) в A ; в частности, последовательность $\alpha: \{j\} \rightarrow A: j \mapsto a$ именуем словом длиной в один ($|\alpha| = 1$).

Для краткости пишут $\alpha = a_{i_1} a_{i_2} \dots a_{i_l}$, в частности, $\alpha = a$. Следует подчеркнуть, что слово есть множество упорядоченных пар $j \mapsto a_{i, j+1}$ ($j=0, 1, \dots, l-1$), или же произведение слов длиной в один, а не приписывание одной буквы к другой.

Определение 4: Пустым словом ε в любом алфавите A называется пустая последовательность $\varepsilon: \emptyset \rightarrow A$ нулевой длины ($|\varepsilon| = 0$), т.е. $\varepsilon = \emptyset$.

Объем понятия «пустое слово» – это пустое множество, его содержание представляет собой слово длиной ноль и пустое слово является нейтральным элементом по отношению к операции произведения слов. Пустое слово (Λ) по [3, с.9.] значит, что нет внешних воздействий на преобразователь. У условимся считать, что символ a_0 внешнего алфавита A является «пустым». Наличие [6, с.148-149] пустого символа (Λ) в некоторой ячейке (например, ленты машины Тьюринга) содержательно означает, что в ней ничего не записано. С одной стороны, символ m_0 – специальный знак пробела между словами [7, с.258], и с другой [7, с.259], тот же самый символ m_0 играет роль пустого символа, причем символом m_0 (буквой) подразумевается элемент алфавита [7, с.258]. Пусть $A1$ – алфавит и $\lambda \notin A1$ [8, с.511]: «Назовем λ пустым символом (пробелом), а $A2 = A1 \cup \{\lambda\}$ – алфавитом с пустым символом.» Напротив, в [9, с.240], говорится: «Предполагается, что все ячейки (ленты машины Тьюринга) заполнены буквами, т.е. совсем пустых ячеек нет. В то же время удобно иметь символ, помечающий ячейку как незаполненную – букву # (пробел), которая обязательно присутствует в ленточном алфавите».

Резюмируя, пустое слово является или не является буквой алфавита, играет или не играет роль пробела и пробел входит или не входит в алфавит.

Теорема 1: Пустое слово ε – это семиотическая модель «ничто».

Доказательство: Действительно, пустое слово ε не содержит никакую конкретную пару $j \mapsto a$ в любом алфавите.

Как ни странно, хотя слово ε именуем пустым, оно не является пустым понятием; его содержание не противоречиво и его объем не пуст. Кроме того, пробел (\sqcup) – это регулярная буква всякого алфавита, хотя, как правило, не публикуется. Очевидно $\varepsilon \notin A$, а $\varepsilon \in A^*$, $\sqcup \in A$, а $\sqcup \notin A^*$, т.е. $\sqcup \neq \varepsilon$.

Множество всех слов A^* над алфавитом A , генерируемых алфавитом A , и свободных от A , является объединением

$$A^* = A^\emptyset \cup A^{\{v\}} \cup A^{\{v, v+1\}} \cup \dots \cup A^{\{v, v+1, \dots, \kappa-1\}} \cup \dots, \text{ где } A^\emptyset = \{\emptyset \rightarrow \rightarrow A\} = \{\varepsilon\},$$

$$A^{\{v\}} = \{\{v\} \mapsto A: v \mapsto a\}, A^{\{v, v+1\}} = \{\{v, v+1\} \mapsto A: v \mapsto a_{i1}, v+1 \mapsto a_{i2}\}, \dots,$$

$$A^{\{v, v+1, \dots, \kappa-1\}} = \{\{v, v+1, \dots, \kappa-1\} \mapsto A: v \mapsto a_{i1}, v+1 \mapsto a_{i2}, \dots, \kappa-1 \mapsto a_{i\kappa}\}, \dots \text{ при } v, \kappa \in \mathbf{N}_0.$$

Напомним, что $A \neq A^{\{v\}}$, и что $\langle A^*, \bullet, \varepsilon \rangle$ – свободный (от A) моноид, где \bullet – оператор конкатенации слов.

Определение 5: Операция произведения (конкатенации) слов в A – это операция $A^* \times A^* \rightarrow \rightarrow A^*$: $\langle \alpha, \beta \rangle \mapsto \alpha\beta$ такая, что, если $\alpha = \{0 \mapsto a_{i1}, 1 \mapsto a_{i2}, \dots, l-1 \mapsto a_{il}\}$ и $\beta = \{0 \mapsto a_{j1}, 1 \mapsto a_{j2}, \dots, k-1 \mapsto a_{jk}\}$, то $\alpha\beta = \{0 \mapsto a_{i1}, 1 \mapsto a_{i2}, \dots, l-1 \mapsto a_{il}, l \mapsto a_{j1}, l+1 \mapsto a_{j2}, \dots, l+k-1 \mapsto a_{jk}\}$.

Слово длиной l в алфавите A может быть определено как упорядоченная l -ка $\langle a_{i1}, a_{i2}, \dots, a_{il} \rangle (\langle a_{i1}, a_{i2}, \dots, a_{il} \rangle \in A^l)$ рекурсивно:

- $\langle a_{i1}, a_{i2} \rangle$ – упорядоченная пара ($\langle a_{i1}, a_{i2} \rangle \in A^2$);
- если $\langle a_{i1} \langle a_{i2}, \dots, a_{i3}, \dots, a_{il} \rangle \rangle$ – упорядоченная пара ($\langle a_{i1}, \langle a_{i2}, a_{i3}, \dots, a_{il} \rangle \rangle \in A^1 \times A^{l-1}$), то $\langle a_{i1}, a_{i2}, \dots, a_{il} \rangle$ упорядоченная l -ка ($\langle a_{i1}, a_{i2}, \dots, a_{il} \rangle \in A^l$);
- ничто другое не есть упорядоченная l -ка.

Упорядоченная пара $\langle a_{i_1}, a_{i_2} \rangle$ вводится, с одной стороны, неявно: $\langle a_{i_1}, a_{i_2} \rangle = \langle a_{i_3}, a_{i_4} \rangle \Leftrightarrow a_{i_1} = a_{i_3} \& a_{i_2} = a_{i_4}$ и, с другой, в явном виде по Винеру $\{\{\emptyset, \{a_{i_1}\}\}, \{\{a_{i_2}\}\}\}$, Куратовскому $\{\{a_{i_1}\}, \{a_{i_1}, a_{i_2}\}\}$, Хаусдорфу $\{\{a_{i_1}, 1\}, \{a_{i_2}, 2\}\}$ и т.п. Остается доопределить упорядоченные «единицу» $\langle \langle a \rangle \rangle$ и «ноль» $\langle \langle \rangle \rangle$. Введем неупорядоченную l -ку $(a_{i_1}, a_{i_2}, \dots, a_{i_l}) = \{a_{i_1}, a_{i_2}, \dots, a_{i_l}\}$; тогда естественно считать, что $\langle a \rangle = (a) = \{a\}$ и $\langle \rangle = \{ \} = \emptyset$. К сожалению, $A^l = A^1 \times A^{l-1}$ ($l \geq 1$), причем $A^1 = A$ *, т.е. если $\langle a \rangle \in A^1$ то $\langle a \rangle \in A$ и тем самым $\langle a \rangle = a$, что не приемлемо, т.к. буква a , генерирующая слово $\langle a \rangle$ длиной один, не совпадает с вырабатываемым словом.

Теорема 2: Существует только одно пустое ε слово в любом алфавите A , для которого, если α слово в A , то $\varepsilon\alpha = \alpha\varepsilon = \alpha$. (Доказательство следует из определения 4 от противного с учетом определения 5).

5. КОНЕЧНЫЙ АВТОМАТ

Пусть \mathbf{R}_0^+ – множество неотрицательных вещественных чисел, $t, t + \Delta\tau$ – моменты времени ($t, t + \Delta\tau \in \mathbf{R}_0^+, \Delta\tau > 0$) и $[t, t + \Delta\tau)$ – промежуток времени ($[t, t + \Delta\tau) \subset \mathbf{R}_0^+$). Функция $[t, t + \Delta\tau) \rightarrow \mathbf{N}_0 : t \mapsto v, t + \Delta\tau \mapsto v + 1$ кодирует моменты времени $t, t + \Delta\tau$ натуральными числами $v, v + 1$. Модифицируя входное слово, «состояние» и выходное слово длиной в один, т.е. $\{[v, v + 1)\} \rightarrow X : [v, v + 1) \mapsto x^{v, v+1}, \{v\} \rightarrow S : v \mapsto s^v, \{[v, v + 1)\} \rightarrow Y : [v, v + 1) \mapsto y^{v, v+1}$, получим множество соответственно X, S, Y слов длиной в один.

Аналогично, пусть X^*, Y^* суть соответственно множества входных и выходных слов вида ($\kappa = 1, 2, \dots$) ($\{s\} \neq \emptyset; \langle a, s \rangle \in A \times A^0$ с тем, что $\langle a, s \rangle = a$ и s – любой символ, а не $A^0 = \emptyset$, ибо $A = A \times A^0$.)

$$\begin{aligned} \{[v, v + 1), [v + 1, v + 2), \dots, [v + \kappa - 1, v + \kappa)\} &\rightarrow X : [v, v + 1) \mapsto x^{v, v+1}, [v + 1, v + 2) \mapsto x^{v+1, v+2}, \dots, \\ [v + \kappa - 1, v + \kappa) &\mapsto x^{v+\kappa-1, v+\kappa} (= x^{v, v+1} x^{v+1, v+2} \dots x^{v+\kappa-1, v+\kappa}), \\ \{[v, v + 1), [v + 1, v + 2), \dots, [v + \kappa - 1, v + \kappa)\} &\rightarrow Y : [v, v + 1) \mapsto y^{v, v+1}, [v + 1, v + 2) \mapsto y^{v+1, v+2}, \dots, \\ [v + \kappa - 1, v + \kappa) &\mapsto y^{v+\kappa-1, v+\kappa} (= y^{v, v+1} y^{v+1, v+2} \dots y^{v+\kappa-1, v+\kappa}). \end{aligned}$$

Определение 6: Конечный детерминированный автомат Мили задается как упорядоченная шестерка

$$(1) \quad \mathcal{A} = \langle X, S, Y, \delta, \lambda, s^0 \rangle,$$

где X, S, Y – алфавит соответственно входов, состояний, выходов, δ, λ функции соответственно переходов $\delta : S \times X \rightarrow S : \langle s^v, x^{v, v+1} \rangle \mapsto s^{v+1}$, выходов $\lambda : S \times X \rightarrow Y : \langle s^v, x^{v, v+1} \rangle \mapsto y^{v+1}$ и s^0 – начальное состояние ($s^0 \in S$) [2, 10, 11, 12].

Автомат (1) можно рассматривать как модель динамического, стационарного (т.е. с независимыми от времени свойствами) логического преобразователя, который, находясь в момент времени v в исходном состоянии s^v и получая входное воздействие $x^{v, v+1}$ на промежутке времени $[v, v + 1)$, оказывается в момент времени $v + 1$ в, предсказанном (!) в момент времени v , состоянии – преемнике s^{v+1} и реагирует на промежутке времени $[v, v + 1)$

откликом $y^{v,v+1}$. Еще раз, в записи $\delta(s^v, x^{v,v+1}) = s^{v+1}$ не является s^{v+1} состоянием из будущего момента времени $v+1$ относительно момента времени v , а своим актуальным (в момент времени v) предсказанием.

Мы можем непосредственно продолжить функции δ, λ на X^*, Y^* :

$$\begin{aligned} \delta^* : S \times X^* \rightarrow S : \langle s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa} \rangle &\mapsto s^{v+\kappa}, \\ \lambda^{**} : S \times X^* \rightarrow Y^* : \langle s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa} \rangle &\mapsto y^{v,v+1} y^{v+1,v+2} \dots y^{v+\kappa-1,v+\kappa}, \end{aligned}$$

в частности,

$$\lambda^* : S \times X^* \rightarrow Y : \langle s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa} \rangle \mapsto \text{proj}_{v+\kappa-1} y^{v,v+1} y^{v+1,v+2} \dots y^{v+\kappa-1,v+\kappa} = y^{v+\kappa-1,v+\kappa},$$

определив эти функции с помощью композиций (\circ - оператор композиции) функций

$(S \times X^* \rightarrow S) = (S \times X \rightarrow S) \circ (S \times X^* \rightarrow S)$, т.е. $\delta^*(\delta(s^v, x^{v,v+1}), x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa}) = s^{v+\kappa}$ с тем, что $\delta(s^v, x^{v,v+1}) = \delta^*(s^v, x^{v,v+1})$,

$$(S \times X^* \rightarrow Y^*) = ((S \times X^* \rightarrow S) \circ (S \times X \rightarrow Y))^*,$$

т.е. $\lambda^{**}(s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa}) = \lambda(\delta^*(s^v, \square_{-,v}), x^{v,v+1}) \lambda(\delta^*(s^v, x^{v,v+1}), x^{v+1,v+2}) \dots$

$$\lambda(\delta^*(s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-2,v+\kappa-1}), x^{v+\kappa-1,v+\kappa}) = y^{v,v+1} y^{v+1,v+2} \dots y^{v+\kappa-1,v+\kappa},$$

$$(S \times X^* \rightarrow Y) = (S \times X^* \rightarrow S) \circ (S \times X \rightarrow Y),$$

т.е.

$$\lambda^*(s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa}) = \lambda(\delta^*(s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-2,v+\kappa-1}), x^{v+\kappa-1,v+\kappa}) = y^{v+\kappa-1,v+\kappa};$$

иными словами

$$\begin{aligned} \lambda^{**}(s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa}) &= \lambda^*(s^0, x^{v,v+1}) \lambda^*(s^v, x^{v,v+1} x^{v+1,v+2}) \dots \\ &\dots \lambda^*(s^v, x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa}), \end{aligned}$$

причем $\text{proj}_{v+\kappa-1}$ – проекция на ось $(v+\kappa-1)$.

Обозначим для простоты $x^{v,v+1} x^{v+1,v+2} \dots x^{v+\kappa-1,v+\kappa}$ и $y^{v,v+1} y^{v+1,v+2} \dots y^{v+\kappa-1,v+\kappa}$ соответственно через $\xi^{v,v+\kappa}$ и $\nu^{v,v+\kappa}$. По Теореме 2. имеет место $\varepsilon \xi^{v,v+\kappa} = \xi^{v,v+\kappa} \varepsilon = \xi^{v,v+\kappa}$ и $\varepsilon \nu^{v,v+\kappa} = \nu^{v,v+\kappa} \varepsilon = \nu^{v,v+\kappa}$. Отсюда естественно $\delta^*(s^v, \xi^{v,v+\kappa}) = \delta^*(s^v, \xi^{v,v+\kappa} \varepsilon) = \delta(s^v, \xi^{v,v+\kappa}) = s^{v+\kappa}$ и $\lambda^{**}(s^v, \xi^{v,v+\kappa}) = \lambda^{**}(s^v, \xi^{v,v+\kappa} \varepsilon) = \lambda^{**}(s^v, \xi^{v,v+\kappa}) = \nu^{v,v+\kappa}$, в частности, $\lambda^*(s^v, \xi^{v,v+\kappa}) = \lambda^*(s^v, \xi^{v,v+\kappa} \varepsilon) = \lambda^*(s^v, \xi^{v,v+\kappa}) = y^{v+\kappa-1,v+\kappa}$.

Но, например в [3, с.17; 12, с.218] приводится, что $\delta^*(s^v, \varepsilon) = s^v, \lambda^*(s^v, \varepsilon) = \varepsilon$, хотя в [3, с. 9] находим: пустое слово значит, что нет внешних воздействий на автомат. Действительно, $\delta^*(s^v, \xi^{v,v+\kappa}) = \delta^*(\delta^*(s^v, \varepsilon), \xi^{v,v+\kappa}) = \delta^*(s^v, \xi^{v,v+\kappa})$; отсюда $\delta^*(s^v, \varepsilon) = s^v$. Также $\lambda^{**}(s^v, \xi^{v,v+\kappa}) = \lambda^*(s^v, \varepsilon^{o,v}) \lambda^{**}(\delta^*(s^v, \varepsilon), \xi^{v,v+\kappa}) = \varepsilon \lambda^{**}(s^v, \xi^{v,v+\kappa}) = \lambda^{**}(s^v, \xi^{v,v+\kappa})$; отсюда $\lambda^*(s^v, \varepsilon) = \varepsilon$. Напротив, «пустая» последовательность ε обладает тем свойством, что произведение любой последовательности α на ε дает слова $\alpha\varepsilon = \varepsilon\alpha = \alpha$, т.е. на вход автомата не подается ни $\alpha\varepsilon$, ни $\varepsilon\alpha$, а α [11, с.227]. Введение «пустой» последовательности удобно при

записи регулярных выражений, где «пустая» последовательность не выступает как входное или выходное слово автомата [13, с.227].

Как показано выше, можно обобщить функции δ, λ , не прибегая к пустому слову, ибо, несомненно, нельзя «отсутствие непустого слова» подать на вход автомата; тем более, что вряд ли можно автомат заставить вырабатывать «отсутствие непустого слова». Но можно предполагать, что автомат виртуально реагирует на пробел, т.е. $\delta^*(s^0, \sqcup^{0,v}) = s^0$ и пробел вырабатывает, т.е. $\lambda^{**}(s^0, \sqcup^{0,v}) = \sqcup^{0,v}$.

6. НЕНАБЛЮДАЕМЫЕ СЛОВО И ВОЗМУЩЕНИЕ

Пусть задан полуавтомат

$$(1) \quad \mathcal{A} = \langle X, S, \delta, s^0 \rangle,$$

в котором все его состояния достижимы из s^0 , т.е. $(\forall s^v \in S)(\exists \xi^{0,v} \in \mathcal{X}^*)(\delta^*(s^0, \xi^{0,v}) = s^v)$. Далее пусть на X^* задана право инвариантная эквивалентность (правая конгруэнтность) $\xi^{0,v} \equiv_N \zeta^{0,v} (\xi, \zeta, \eta \in \mathcal{X}^*)$ [10,12] тогда и только тогда, когда

$$\delta^*(s^0, \xi^{0,v}) = \delta^*(s^0, \zeta^{0,v}) = s^v \Rightarrow \delta^*(s^v, \xi^{0,v} \eta^{v,v+\kappa}) = \delta^*(s^v, \zeta^{0,v} \eta^{v,v+\kappa}) = s^{v+\kappa},$$

определяющая на X^* конечное разбиение

$$(\mathcal{X}^*/\equiv_N) = \bigcup_{s^v \in S} [\xi^{0,v}]_{s^v},$$

классы которого $[\xi^{0,v}]_{s^v} = \{ \xi^{0,v} \in \mathcal{X}^* \mid \delta^*(s^0, \xi^{0,v}) = s^v \}$ располагают избранными входными словами в качестве своих представителей. Тогда определяем полуавтомат Нерода [13,14]

$$(2) \quad \langle X, (\mathcal{X}^*/\equiv_N), \Delta, [\varepsilon] \rangle,$$

где Δ – функция переходов $\Delta: (\mathcal{X}^*/\equiv_N) \times X \rightarrow (\mathcal{X}^*/\equiv_N)$: $\langle [\xi^{0,v}]_{s^v}, x^{v,v+1} \rangle \mapsto [\xi^{0,v} x^{v,v+1}]_{s^{v+1}} (= [\xi^{0,v+1}]_{s^{v+1}})$ с тем, что, если $[\xi^{0,v}]_{s^v} = [\zeta^{0,v}]_{s^v} \Rightarrow \Rightarrow [\xi^{0,v+1}]_{s^{v+1}} = [\zeta^{0,v+1}]_{s^{v+1}}$, то $\Delta([\xi^{0,v}]_{s^v}, x^{v,v+1}) = [\xi^{0,v+1}]_{s^{v+1}} = \Delta([\zeta^{0,v}]_{s^v}, x^{v,v+1}) = [\zeta^{0,v+1}]_{s^{v+1}}$

■

Поскольку не известна входная предыстория полуавтомата (2), т.е. не известны входные слова, переводившие автомат (2) в его начальное состояние s^0 , и входная предыстория в начальном состоянии существует в накопленном виде, постольку входная предыстория ненаблюдаемая. Таким образом, вполне оправдано обозначить ее через символ ε . Аналогично дела обстоят в случае любой входной истории того или иного состояния. Накопленная, «неразборчивая» форма входных предыстории и историй объясняется отсутствием стирания «содержимого» состояний автомата.

Преобразователь и его окрестность вырабатывают массовые, однородные, относительно независимые, случайные возмущения. Случайность воздействия возмущений на преобразователь неустранима и представляет собой неопределенность вида физической неоднозначности. Определение причин случайности не является ни существенным, ни

нужным и принципиально не осуществимо. В данном рассмотрении ограничимся неизмеримыми возмущениями.

Определение 7. Эпсилон-полуавтоматом называется упорядоченная четверка [15]

$$(3) \quad \mathcal{A}_\varepsilon = \langle X \cup \{\varepsilon\}, S, \delta_\varepsilon, s^0 \rangle,$$

где δ_ε – функция переходов $\delta_\varepsilon = \mathcal{S} \times (\mathcal{X} \cup \{\varepsilon\}) \rightarrow \mathcal{S} : \langle s^v, x^{v,v+1} \rangle \mapsto s^{v+1}, \langle s^v, \varepsilon \rangle \mapsto s^v$ и $\langle s^v, \varepsilon \rangle \mapsto s^{v+1}$ ($s^v \neq s^{v+1}$ при воздействии неизмеримого возмущения) – ε - переход.

Теорема 3.: Из эпсилон-автомата можно исключить его ε -переходы.

Доказательство: Введем ε -замыкание состояния s^v : ε -closure (s^v) = $\delta_\varepsilon(s^v, \varepsilon)$. Тогда для автомата без ε -переходов (2) имеет силу $s^{v+1} = \delta(s^v, x^{v,v+1}) = \bigcup_{\hat{s}^v} \delta_\varepsilon(\hat{s}^v, x^{v,v+1})$, где $\hat{s}^v = \varepsilon$ -closure(s^v). □

Нам не известен ни способ, ни момент времени возникновения возмущений. Если неизмеримое возмущение воздействует на (4), находящемся в s^v то (целенаправленный стимул $x^{v+1,v+2}$ вызовет переход в s^{v+2} ; но, если неизмеримое возмущение не воздействует на (3), то стимул $x^{v,v+2} = x^{v,v+2}$ подается на вход, находящемся в s^{v+1} , а не в s^v ?

7. ЗАКЛЮЧЕНИЕ

Автор надеется, что удалось убедительно показать несостоятельность воздействия пустого слова на автомат и выдачи пустого слова автоматом. Ведь в случае надобности вместо пустого слова имеется слово с пробелом.

Следует отметить опасность толкования сокращений; ведь слово не есть буква и не состоит из букв, а представляет собой произведение слов длиной в один.

Поскольку символ ε зарезервирован для ненаблюдаемого пустого слова, постольку может стать опасным обозначать через ε хотя неизмеримые, но непустые возмущения.

ЛИТЕРАТУРА

1. Кирилов В.И., Старченко А.А. Логика. – М.: Высшая школа, 1962.
2. Глушков В.М. Синтез цифровых автоматов. – М.: ГИФМЛ, 1962.
3. Курдавцев В.Б., Алешин С.В., Подколзин А.С. Введение в теорию автоматов. – М.: Наука, 1985.
4. Harrison M.A. Introduction to Switching and Automata Theory - New York - ... - Sydney: Mc Graw – Hill Book Co., 1965.
5. Hopcroft J.E., Ullman J.D. Formálne jazyky a automaty. – Bratislava: Alfa, 1975, перевод с английского Rován V., Mikulecký P.
6. Шоломов Л.А. Основы теории дискретных логических и вычислительных устройств. - С.П.,М., Краснодар: изд. Лань, 2011, 3-е изд., ISBN 978-5-8114 – 1197-9
7. Горбатов В.А. Фундаментальные основы дискретной математики. Информационная математика. – М.: Наука, 1999, ISBN 5-02-015238-2
8. Иванов В.А., Медведев В.С. Математические основы теории оптимального и логического управления. – М.: изд. МГТУ, 2011, ISBN 978-5-7038-3366-7

9. Крупский В.И., Плиско В.Е. Математическая логика и теория алгоритмов. – М.: Академия, 2013, ISBN 978-5-7695-9559-2
10. Ершов Ю.Л., Палютин Е.А. Математическая логика. – М.: Наука, 1979
11. Крупский В.И., Плиско В.Е. Математическая логика и теория алгоритмов. – М.: Академия, 2013, ISBN 978-5-7695-9559-2
12. Трахтенброт Б.А., Барздинь Г.М. Конечные автоматы (Поведение и синтез) – М.: Наука, 1970.
13. Калман Р., Фалб М., Арбиб М. Очерки по математической теории систем. – М.: Мир, 1971,. перевод с английского Наппембаума Э.Л.
14. Айзерман М.А., Гусев Л.А., Розоноэр Л.И., Смирнова И.М., Таль А.А. Логика, автоматы, алгоритмы. – М.: ГИФМЛ, 1963.
15. D'Souza D., Shankar P. Modern Applications of Automata Theory. – New Jersey - ... - Chennai: World Scientific Publishing Co., 2012, ISBN – 13-978-981-4271-04-2

Работа поступила 18.06.2018г.



АВТОРЫ

Войтов Владимир Кузьмич

Кандидат технических наук, профессор кафедры прикладной информатики и мультимедийных технологий факультета информационных технологий МГППУ
vvoi@mail.ru

Бокр Йозеф

Западнечешский университет в г. Пльзене (Чехия)
bokr@kiv.zcu.cz

Степанов Михаил Евграфович

кандидат педагогических наук, доцент кафедры прикладной математики факультета информационных технологий МГППУ
mestepanov@yandex.ru