Бухвалова Вера Вацлавовна,
Зациорский Артем Сергеевич

УДК 004.434

ЯЗЫК DROL КАК СРЕДСТВО РАСШИРЕНИЯ ГРАФИЧЕСКИХ ВОЗМОЖНОСТЕЙ СИСТЕМЫ LaTeX

Аннотация

В статье описана реализация геометрического языка спецификаций **DROL,** предназначенная для описания геометрических построений и записи алгоритмов решения геометрических задач на плоскости. Предлагается также использовать этот язык для создания банков типовых рисунков для учебных курсов, презентаций и публикаций. Препроцессор **DROL2TeX** преобразует программу на языке **DROL** в набор команд графического пакета **TikZ,** результатом выполнения которых служит получение запрограммированного изображения. Расширение графических возможностей системы LaTeX демонстрируется на примерах.

Ключевые слова: геометрический язык спецификаций, язык **DROL**, LaTeX, ООП, графические пакеты..

1. ВВЕДЕНИЕ

Эта статья предназначена как для тех, кто уже предпочитает готовить различные материалы (статьи, презентации, оригинал-макеты книг и т. п.) в системе LaTeX, так и для тех, кто уже попробовал работать в этой системе, но столкнувшись с большими трудностями при создании даже простых рисунков, предпочел другое программное обеспечение. Начнем с того, что поясним, с чем связаны эти трудности, а затем опишем, как они могут быть преодолены с помощью языка DROL и препроцессора DROL2TeX.

Напомним, что в системе LaTeX для получения простых графических изображений, состоящих из отрезков, окружностей, овалов и кривых Безье, имеется окружение `picture`[1] и небольшой набор команд, с помощью которых задаются элементы изображений. Для расширения возможностей окружения picture создано довольно много графических пакетов (см. [8]). Однако при использовании команд из этих пакетов, как и при использовании окружения picture, необходимо явно задавать координаты точек, которые определяют элементы изображения.

Другой подход к расширению графических возможностей системы LaTeX заключается в создании препроцессора с графического языка в набор команд этой системы, возможно, расширенный каким-либо

[1] Что касается языка TeX, то для него картинка – это просто прямоугольник, который надо разместить на странице. От пользователя нужны только размеры этого прямоугольника. Команда `\special` предоставляет возможность вставить практически любой фрагмент (в частности, графический) в набираемый текст. Отображением картинки на страницу занимается драйвер.

графическим пакетом. Главное отличие такого подхода – возможность явно не указывать координаты точек, а задавать только их положение относительно друг друга (выше, ниже, пересечение и т. п.). Пример реализации этого подхода описан в работе [10]: препроцессор с графического языка Diag в набор команд графического пакета PiCTeX.

В нашей статье описана версия препроцессора с языка DROL в набор команд пакета TikZ ([11]). Выбор авторами для последних версий препроцессора именно этого пакета связан с его широкими графическими возможностями и надежной работой с системой LaTeX. Впрочем, пользователю не обязательно уметь работать с этим пакетом – достаточно только добавить команду `\usepackage{tikz}` в преамбулу своего файла.

Работа имеет следующую структуру: настоящее введение, общая информация о языке DROL и его реализациях, описание основных элементов языка с примерами, пример создания серии связанных рисунков, список литературы. Все рисунки в статье были созданы с помощью языка DROL и препроцессора DROL2TeX (v. 3.2.2).

2. ЯЗЫК СПЕЦИФИКАЦИЙ DROL

Идея создания языка DROL (DRawing Oriented Language) как предметно-ориентированного языка для описания структуры изображений принадлежала И.В. Романовскому. Первая версия языка (1972 г.) представляла из себя расширение геометрическими типами и операциями языка Algol 68. Окончательным результатом работы программы на языке DROL являлся специальный код описанного в ней изображения. Само изображение с помощью специальной программы можно было получить на графопостроителе. Язык использовался для записи алгоритмов элементарной геометрии и описания выкроек одежды ([1]). Отметим, что и в настоящее время продолжаются работы по созданию геометрических предметно-ориентированных языков (см., например, [6]).

Все дальнейшие изменения в языке DROL были связаны с расширением области его применения. Если первоначально язык был средством описания несложных чертежей с элементами вычислений, то далее он стал языком спецификации для записи алгоритмов вычислительной геометрии на плоскости, сохранив при этом свое имя. На выбор структуры и синтаксиса языка наибольшее влияние оказали такие языки программирования (ЯП) как Algol 68, CLU. Основная цель, которая ставилась при этом, – поддержка технологий структурного, сборочного и объектно-ориентированного программирования и использование всех видов абстракций. Синтаксис языка не полностью формализован и прежде всего в тех конструкциях, которые имеют разный синтаксис даже в популярных ЯП. В таких случаях разрешена любая наглядная форма записи. Примерами таких конструкций являются индексный список в операторе цикла с параметром и комментарий (см. [3]).

Одновременно продолжалось создание реализаций подмножеств языка DROL, ориентированных на решение различных классов задач. Например, версия DROL для задач прямоугольного раскроя была основана на языке Turbo Pascal с использованием библиотеки Turbo Vision ([2]). Эту реализацию следует признать удачной, так как созданная на ее базе программа прямоугольного раскроя до сих пор находится в рабочем состоянии (более 20 лет!) и успешно используется для тестирования вновь создаваемого программного обеспечения.

Принципиально от описанных выше реализаций языка DROL отличается подход, при котором создается интерпретатор, непосредственно выполняющий все необходимые вычисления по алгоритмам. Интерпретатор не привязан к какому-либо ЯП и записывает всю информацию об имеющихся в программе объектах в собственном внутреннем коде. Далее этот код может быть использован для генерации всего изображения или его фрагмента на каком-либо ЯП. На данном этапе в качестве ЯП был выбран язык LaTeX. К настоящему моменту под руководством В.В. Бухваловой её ученики создали уже несколько версий интерпретатора. Начиная с первой версии (В.В. Голубев, 1998), интер-

претатор реализован как многооконный редактор текстов, соответствующий международному стандарту SAA/CUA (System Application Architecture Common User Access) и поддерживает стандартный набор операций работы с файлами, блоками и буфером обмена. Имеется возможность редактировать исходный текст, запускать процесс интерпретации и просматривать ее результаты, не выходя из оболочки. В 2001 г. В.О. Воронков переписал интерпретатор на языке Delphi 6.0 для работы в системе Windows.

Аспирант А.С. Зациорский внес существенные изменения в интерпретатор ([4,5]). Устаревшие графические пакеты он заменил одним современным и мощным – TikZ ([9]), что позволило значительно улучшить визуальное качество изображения и устранить проблемы совместимости. Был создан WYSIWYG-пред-просмотрщик для проверки семантической корректности описания изображения без компиляции LaTeX-документа. Появилась возможность задавать цвет изображаемых объектов и их текстовых подписей. Набор выходных форматов пополнился растровыми форматами BMP, JPG и PNG. Был улучшен пользовательский интерфейс (моноширинные шрифты, возможность автоматического создания осей координат, помещение сгенерированной LaTeX-программы в буфер обмена для дальнейшей вставки в документ и т. д.). Расширена библиотека графических функций и операций. Добавление условного оператора и оператора Exit позволило объединять описания серии рисунков с единой элементной базой (см. раздел 4).

Сам программный продукт (интерпретатор DROL2LaTeX, v. 3.2) и подробное руководство к нему с многочисленными примерами размещено авторами в свободном доступе на сайте exponenta.ru ([12]).

3. ОБЗОР ЯЗЫКА DROL

В этом разделе приведен краткий обзор языка DROL с примерами. Более подробное описание языка можно найти в [3]. При этом следует иметь в виду, что DROL, как и положено языку программирования при расширении области его применения, постоянно развивается, предоставляя пользователю все новые возможности. При этом соблюдается принцип совместимости – корректность программ, написанных на более ранних версиях языка. В настоящий момент последней является версия 3.2.2.

Словарь языка DROL состоит из основных (латинские буквы, цифры) и специальных символов. К специальным символам относятся знаки операций и разделители. Набор знаков операций и знаковых разделителей заимствован из языка Pascal. Зарезервированные слова, относящиеся к геометрическим типам данных и операциям над ними, являются сокращениями от соответствующих английских слов.

Для именования объектов в программе используются идентификаторы. Идентификаторы могут быть константами, которые ссылаются в процессе работы программы на один объект, или переменными, которые могут ссылаться на разные объекты. Переменные создаются с помощью описаний и имеют типы, указанные в этих описаниях. Язык DROL является строго типизированным – все идентификаторы, встречающиеся в программе, являются либо зарезервированными, либо явно описаны. Область действия описания заключена между самим описанием и концом программы. Разрешено объединять описания переменных с присваиванием им начальных значений и размещать описания объектов в любом месте программы до первого их использования. Идентификаторы регистронезависимы. Например, **`point`** и **`Point`** ссылаются на один и тот же тип. Комментарии до конца строки могут быть заданы с помощью символа %.

3.1. Типы данных и встроенные константы

Препроцессор **DROL2TeX 3.2** поддерживает следующие типы данных (перечисление по алфавиту): **`angle`** – угол, **`arc`** – дуга окружности, **`bool`** – булевский тип, **`circle`** – окружность, **`direction`** – направление, **`line`** – прямая, **`lineseg`** – отрезок,

orientation – ориентация (по или против часовой стрелки), **point** – точка, **ray** – луч, **real** – вещественный тип, **rect** – прямоугольник, **spline** – естественный кубический сплайн. Предопределены следующие встроенные константы:

– **orientation Clw, Cclw** ; % ориентация по (против) часовой стрелке;

– **point Origin, Infinity** ; % начало координат и бесконечно удалённая точка (в ней пересекаются параллельные прямые);

– **real Pi** ; % число π;

– **line XAxis, YAxis** ; % оси абсцисс и ординат;

– **bool True, False** ; % логические значения «истина» и «ложь».

3.2. Структура программы

Программа на языке DROL имеет следующую структуру (см. листинг 1).

Таким образом, DROL-программа представляет собой определение множества объектов и изображение некоторого подмножества этих объектов.

3.3. Функции и операции

Удобство описания рисунков на языке DROL связано с имеющимися геометрическими типами данных (перечислены выше) и с богатой библиотекой операций[2] и функций, позволяющей описывать геометрические построения в наглядной форме. В этом разделе мы рассмотрим только те операции и функции, которые, как показала практика, используются чаще всего. Полный список доступных операций и функций имеется в [3, 12]. Отметим, что каждый из приведенных далее примеров после удаления номеров строк, которые мы добавили для удобства комментирования, является корректной DROL-программой.

Конструктор. Полиморфный конструктор **Cr** (от create), определен для всех геометрических типов. Ему соответствует простейший способ определения объекта соответствующего типа. Например, в следующих операторах присваивания определяются: точки *A* и *B*, имеющие указанные координаты; отрезок *AB*, соединяющий точки *A* и *B*, и окружность *C* с центром в точке *A* и радиусом, равным 30:

```
point A = Cr(10,10), B = Cr(20,20);
lineseg AB = cr(A, B);
circle C = Cr(A,30);
```

Точки и лучи. Группа унарных операций **Rt**, **Lt**, **Up**, **Dw** определяет луч, который выходит из точки (параметр) вправо, влево, вверх и вниз соответственно. Бинарные операции **Perp**, **Par** определяют луч, который выходит из точки (левый параметр) перпендикулярно и параллельно прямой или отрезку (правый параметр) соответственно. Бинарная операция **On** определяет точку на луче (левый параметр), расстояние до которой от начала луча равно значению правого параметра. Значением функции **Middle** является точка, которая является серединой отрезка, соединяющего две точки (параметры).

Листинг 1

```
Программа = Параметры  |  Оператор,  {Оператор};
Оператор = Прекращение работы  |  Условный оператор  | Присваивание |
           Определение  |  Изображение;
Прекращение работы = «EXIT»,  «;»;
Условный оператор = «IF», Логическое значение,  «THEN»,
                    {Оператор}, [«ELSE», {Оператор}], «FI», «;»;
Присваивание = Идентификатор,  «=», Выражение,  «;»;
Определение = Тип,  Идентификатор,  «=», Выражение,  «;»;
Изображение = «DRAW», «(», Идентификатор, [«:», Сдвиг, «:»,
              Текст], [«@», Цвет], Идентификатор, [«:», Сдвиг,
               «:», Текст], [«@», Цвет], «)», «;»;
```

[2] Идея использовать для обозначения элементарных геометрических построений не только функции, но и там, где это оправдано, унарные и бинарные операции принадлежит И.В. Романовскому.

Листинг 2

```
1:  real x = 20, h = 15; % длины основания и высоты
2:  point A = Origin; % точка A - в начале координат
3:  point C = Rt A On x; % точка C - правее точки A на x
4:  point D = Middle(A, C); % точка D - основание высоты
5:  point B = Up D On h; % точка B - выше точки D на h
6:  lineseg AB = cr(A,B), BC = cr(B,C), AC= cr(A,C), AD = cr(A,D);
7:  % изображение определенных объектов
8: draw(A:r:A, B:b:B, C:l:C, D:t:D, AB, BC, AC, AD);
```

Пример 1. Опишем построение равнобедренного треугольника ABC: заданы основание $AC(x)$ и высота, опущенная на основание (h) (рис. 1).

В листинге 2 приведен текст программы, которая выполняет это построение, и вывод получившегося рисунка.

Комментарий к программе. Описание рисунка обычно начинается с описания его параметров. В данном случае – это длина основания x и длина высоты h (строка 1). Трансляция программы будет выполняться с заданными значениями параметров ($x = 20$, $h = 15$). В строках 2–6 определены элементы рисунка (вершины и стороны треугольника). Изображение объектов осуществляется с помощью полиморфной функции **draw** (строка 8). Заданы текстовые подписи и направление их сдвига.

Дуги и окружности. При работе с окружностями имеется возможность строить касательные. В ряде случаев удобно использовать не окружности, а дуги. Например, функция **CrPoints** определяет дугу заданного радиуса (третий параметр), соединяющую две заданные точки (первый и второй параметры). А функция **Conj** определяет дугу заданного радиуса (третий параметр), которая сопрягает две заданные прямые (первый и второй параметры).

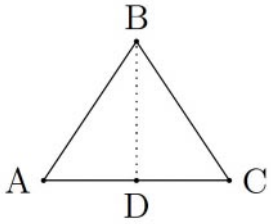

Рис. 1. Равнобедренный треугольник

```
real R =   ... ;
point A=   ...,   B=   ...;
line L=   ...,   M =   ...;
arc V = CrPoints(A, B,R);
arc W = Conj(L,M,R);
```

Пересечения. Для прямых и окружностей определены следующие операции пересечения: **Intersect** – пересечение двух прямых; **LIntersect**, **RIntersect** – правое и левое пересечение прямой (левый параметр) с окружностью (правый параметр); **LCirInt**, **RCirInt** – правое и левое пересечение двух окружностей. Предполагается, что прямая имеет направление **(direction)**, поэтому «право» и «лево» определяются с учетом этого направления. Это относится и к касательным — в этом случае выбор делается относительно прямой, соединяющей центр окружности и точку, из которой проводятся касательные.

Пример 2. Опишем построение окружности C с центром в начале координат и радиусом r, и двух касательных к ней, проведенных из точки A, координаты которой заданы (рис. 2).

В листинге 3 приведен текст программы, которая выполняет это построение и вывод получившегося рисунка.

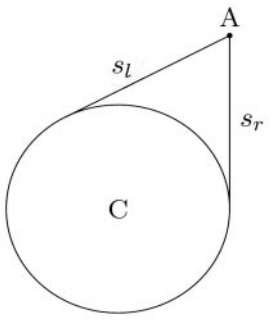

Рис. 2. Окружность и касательные

Листинг 3

```
1:   real r = 15, xa = 15, ya = 25; % радиус и координаты точки
2:   circle C = cr(Origin,r); % определение окружности
3:   point A = cr(xa,ya);   % определение точки A
4:   lineseg sl = A ltangent C, sr = A rtangent C;
5:   % отрезки sl и sr - левая и правая касательные к C из точки A
6:   draw(C:c:C, A:b:A, sl:b:$s_l$, sr:b:$s_r$);
```

Комментарий к программе. Для построения касательных к окружности используются бинарные операции **LTangent** и **RTangent** (строка 4). Если поясняющий текст является формулой (в нашем случае символы с индексом: s_l, s_r), то его необходимо писать, как TeX-формулу (строка 6).

Симметричные объекты. Для получения симметричных объектов всех геометрических типов имеется две полиморфные функции **SymC** (центральная симметрия относительно точки) и **SymA** (симметрия относительно прямой).

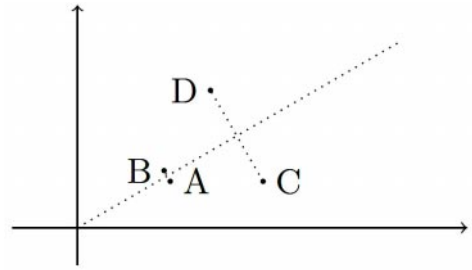

Рис. 3. Симметричные точки

Пример 3. Выполним построение двух пар точек (A, B) и (C, D), симметричных относительно луча, выходящего из начала координат под углом $t°$ (рис. 3).

В листинге 4 приведен текст программы, которая выполняет это построение и вывод получившегося рисунка.

Комментарий к программе. Точка E и отрезок OE определены (строки 4–5) и изображены на рисунке только для того, чтобы сделать его более наглядным. Для получения изображения осей координат достаточно задать соответствующий параметр при трансляции.

Булевские операции и условный оператор. В следующем примере покажем, как использовать условный оператор, булевские геометрические операции и цвета при изображении геометрических объектов.

Пример 4. Выполним построение квадрата R со стороной L. Впишем в него окружность C. Изображение точки $P(x_1, y_1)$ будет зеленым, если эта точка расположена внутри окружности C, и красным – если вне окружности (рис. 4).

Листинг 4

```
 1:  real t = 30, xa = 10, ya = 5, m = 40;
 2:  ray R = Origin dir cr(t);
 3:  % R - луч из начала координат под углом t
 4:  point E = R on 40;
 5:  lineseg OE = cr(Origin, E);
 6:  % OE - изображение луча R
 7:  point A = cr(xa, ya);
 8:  point B = SymA(A, R);
 9:  point C = rt A on xa;
10:  point D = SymA(C, R);
11:  lineseg AB = cr(A,B), CD = cr(C,D);
12:  draw(OE, A:l:A, B:r:B, C:l:C, D:r:D, AB, CD);
```

В листинге 5 приведен текст программы, которая выполняет это построение и вывод получившегося рисунка.

Комментарий к программе. Конструктор **Cr** (строка 3) создает прямоугольник, нижняя левая вершина которого в начале координат, а длины сторон равны. Центр окружности *C* – середина диагонали квадрата. Бинарная логическая операция **inside** вырабатывает значение **True,** если точка *P* расположена внутри окружности *C*. В этом случае изображение точки *P* будет зеленым, в противном случае – красным. При заданных координатах (строка 1) изображение будет зеленым.

4. СОЗДАНИЕ СЕРИИ СВЯЗАННЫХ РИСУНКОВ

Подготовка материалов по многим учебным курсам предполагает включение в них большого количества рисунков. Причем, это касается как курсов, в которых объектами изучения являются геометрические объекты и алгоритмы их обработки (например, машинная графика и вычислительная геометрия), так и курсов, изучающих другие классы объектов, но изложение которых традиционно сопровождается большим числом рисунков (например, исследование операций, микроэкономика и финансы).

В настоящее время процесс создания рисунков остается по-прежнему весьма трудоемким, независимо от того, какие программные средства используются. Поэтому порой авторы копируют рисунки из других источников (не всегда удачно) или «экономят» на рисунках, заменяя их вербальным описанием. В этом разделе на примере алгоритма построения выпуклой оболочки Грэхема показано, как на языке DROL можно создавать единое описание серии связанных единой элементной базой рисунков.

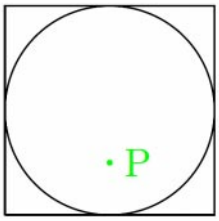

Рис. 4. Вписанная в квадрат окружность и точка *P*

Сначала несколько слов об алгоритме Грэхема (Graham, 1972), который традиционно излагается в базовых курсах по вычислительной геометрии. Этот алгоритм строит выпуклую оболочку *n* точек на плоскости за время $O(n \log n)$ и является, тем самым, оптимальным алгоритмом. Алгоритм включает две фазы: сортировку точек по полярному углу и обход отсортированных точек с анализом угла поворота, образованного тремя текущими точками. В случае поворота влево происходит переход на одну точку вперед, в случае поворота вправо –

Листинг 5

```
 1:  real x1 = 10, y1 = 5, L = 20;
 2:  % координаты точки и длина стороны квадрата
 3:  rect R = Cr(origin, L, L);
 4:  % R - прямоугольник с нижней левой вершиной в начале координат,
 5:  % длины сторон равны L (квадрат)
 6:  point A = Ul R, B = Dr R;
 7:  % вершины R: A - верхняя левая, B - нижняя правая
 8:  circle C = Cr(middle(A, B), L/2),
 9:  % C – окружность, вписанная в квадрат
10:  point P = Cr(x1,y1);
11:  draw(R,C);
12:  if P inside C then % P внутри окружности C
13:   draw(P:l:O@green);
14:  else draw(P:l:O@red);
15:  fi;
```

средняя точка является внутренней точкой и удаляется из списка[3].

Пример 5. Имеется девять точек на плоскости, координаты которых заданы. Опишем девять рисунков, иллюстрирующих по шагам работу алгоритма Грэхема для этого набора точек (рис. 5).

В листинге 6 приведен текст программы, которая выполняет это построение, и вывод

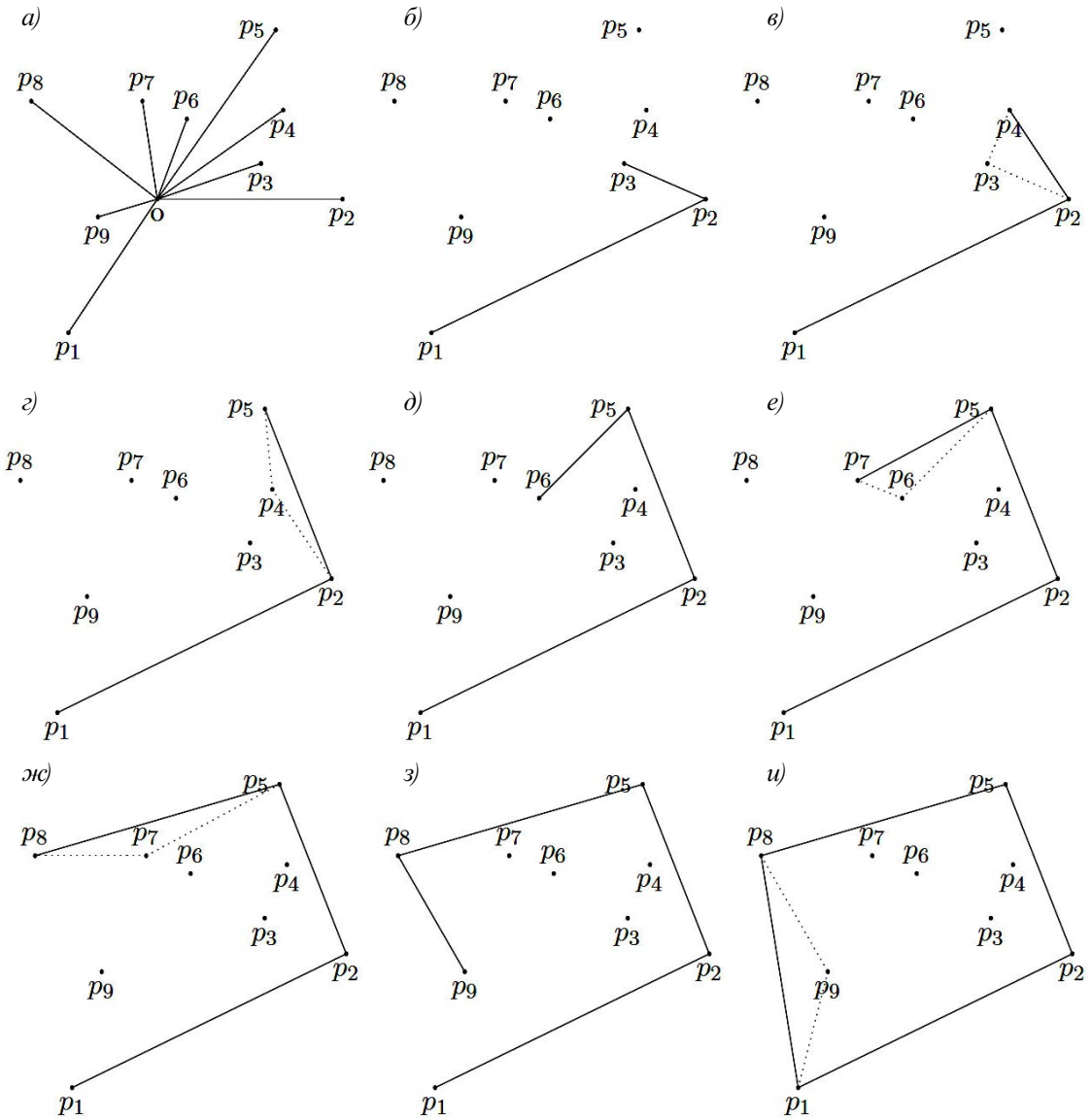

Рис. 5. Шаги алгоритма Грэхема:

а) сортировка точек по углу;
б) угол $p_1p_2p_3$ – левый;
в) угол $p_2p_3p_4$ – правый, исключаем p_3, угол $p_1p_2p_4$ – левый;
г) угол $p_2p_4p_5$ – правый, исключаем p_4, угол $p_1p_2p_5$ – левый;
д) угол $p_2p_5p_6$ – левый;
е) угол $p_5p_6p_7$ – правый, исключаем p_6, угол $p_2p_5p_7$ – левый;
ж) угол $p_5p_7p_8$ – правый, исключаем p_7, угол $p_2p_5p_8$ – левый;
з) угол $p_5p_8p_9$ – левый;
и) угол $p_8p_9p_1$ – правый, исключаем p_9, угол $p_5p_8p_1$ – левый, выпуклая оболочка построена.

[3] За более подробным описанием и анализом этого алгоритма мы отсылаем читателя к учебникам по вычислительной геометрии, например [7].

Листинг 6

```
 1:  real x1 = -12, y1 = -15, x2 = 25, y2 = 0, x3 = 14, y3 = 4,
 2:       x4 = 17, y4 = 10, x5 = 16, y5 = 19, x6 = 4, y6 = 9,
 3:       x7 =-2, y7 = 11, x8 =-17, y8 = 11, x9 =-8, y9 =-2;
 4:  point p1 = cr(x1,y1), p2 = cr(x2,y2), p3 = cr(x3,y3),
 5:        p4 = cr(x4,y4), p5 = cr(x5,y5), p6 = cr(x6,y6),
 6:        p7 = cr(x7,y7), p8 = cr(x8,y8), p9 = cr(x9,y9);
 7:  % индикаторы рисунков
 8:  bool pic1 = False, pic2 = False, pic3 = False,
 9:       pic4 = False, pic5 = False, pic6 = False,
10:       pic7 = False, pic8 = False, pic9 = False;
11:  pic1 = True; %что рисуем
12:  point O = origin; % внутренняя точка выпуклой оболочки
13:  % изображение точек - имеется на всех рисунках
14:  draw(p1:t: '$p_1$', p2:t:'$p_2$', p3:t:'$p_3$', p4:t:'$p_4$',
15:  p5:r:'$p_5$',p6:b:'$p_6$',p7:b:'$p_7$',p8:b:'$p_8$',p9:t:'$p_9$');
16:  % отрезки для рис. 1
17:  lineseg s1 = cr(O, p1), s2 = cr(O, p2), s3 = cr(O, p3),
18:          s4 = cr(O, p4), s5 = cr(O, p5), s6 = cr(O, p6),
19:          s7 = cr(O, p7), s8 = cr(O, p8), s9 = cr(O, p9);
20:  if pic1 then
21:          draw(O:t:o, s1, s2, s3, s4, s5, s6, s7, s8, s9); exit;
22:  fi;
23:  % отрезки для рис.2 — рис.9
24:  lineseg s12 = cr(p1, p2), s23 = cr(p2, p3), s34 = cr(p3, p4),
25:          s24 = cr(p2, p4), s45 = cr(p4, p5), s25 = cr(p2, p5),
26:          s56 = cr(p5, p6), s67 = cr(p6, p7), s57 = cr(p5, p7),
27:          s78 = cr(p7, p8), s58 = cr(p5, p8), s89 = cr(p8, p9),
28:          s91 = cr(p9, p1), s81 = cr(p8, p1);
29:  if pic2 then
30:     draw(s12, s23); exit;
31:  fi;
32:  if pic3 then
33:     draw(s12, s23, s34, s24); exit;
34:  fi;
35:  if pic4 then
36:     draw(s12, s24, s45, s25); exit;
37:  fi;
38:  if pic5 then
39:     draw(s12, s25, s56); exit;
40:  fi;
41:  if pic6 then
42:     draw(s12, s25, s56, s67, s57); exit;
43:  fi;
44:  if pic7 then
45:     draw(s12, s25, s57, s78, s58); exit;
46:  fi;
47:  if pic8 then
48:     draw(s12, s25, s58, s89); exit;
49:  fi;
50:  if pic9 then
51:     draw(s12, s25, s58, s89, s91, s81);
52:  fi;
```

рисунков (фрагментов рис. 5), каждый из которых соответствует шагу алгоритма.

Комментарий к программе. Программа объединяет построение 9 рисунков (по числу шагов алгоритма Грэхема). Каждому из этих рисунков соответствует булевская переменная (строки 8–10). Перед трансляцией программы необходимо в строке 11 указать, какой из рисунков требуется получить (присвоить соответствующей переменной значение **`True`**). После построения этого рисунка процесс трансляции прекращается оператором **`Exit`**. Изображение точек, для которых строится выпуклая оболочка, присутствует на всех рисунках (строки 14–15). Каждому рисунку соответствует свой набор изображаемых отрезков. Точки перенумерованы в порядке возрастания полярного угла. В качестве первой точки (p_1) выбрана нижняя точка, которая заведомо является вершиной выпуклой оболочки. Заметим, что изменение координат заданных точек (при сохранении количества точек и порядка их обхода) не влияет на структуру программы, но может привести к локальному изменению ее текста, начиная со строки 24. Может измениться набор отрезков, которые являются временными или постоянными ребрами выпуклой оболочки заданных точек (строки 24–28), что, в свою очередь, повлияет на набор аргументов функции **`draw`** в строках 33, 36, 39, 42, 45, 48, 51. Даже при изменении количества точек структура программы может быть полностью сохранена, хотя потребуются более значительные изменения в ее тексте.

5. ЗАКЛЮЧЕНИЕ

В статье описано только одно направление развития языка DROL – расширение графических возможностей системы LaTeX. Но процесс создания рисунков *не может не быть трудоемким,* поэтому авторы начали работу по созданию банков типовых рисунков для учебных курсов. О первых результатах сообщается в [5].

Как указывалось ранее, разработанный интерпретатор не привязан к какому-либо ЯП и записывает всю информацию об имеющихся в программе объектах в собственном внутреннем коде. Поэтому в планах авторов добавление возможности сохранения рисунков на языке PostScript, что позволит, в частности, повысить качество получаемых изображений.

Литература

1. *Бухвалова В.В.* Геометрический язык для конструирования одежды // Известия вузов. Технология легкой промышленности, 1986. № 4. С. 132 –138.
2. *Бухвалова В.В.* Использование языка геометрического моделирования в прямоугольном раскрое // Всесоюзная н.-т. конф. «Математическое обеспечение рационального раскроя в САПР»: Материалы конференции. Уфа, 1988. С. 80–87.
3. *Бухвалова В. В.* Задача прямоугольного раскроя: метод зон и другие алгоритмы. СПб.: СПбГУ, 2001.
4. *Бухвалова В.В., Зациорский А.С.* Применение языка DROL для созданияпараметризованных изображений // Обозрение прикладной и промышленной математики, 2011. Т. 18, вып. 5. С. 750–751.
5. *Бухвалова В.В., Зациорский А.С.* Банки типовых рисунков для учебных курсов: применение языка DROL // Обозрение прикладной и промышленной математики, 2012. Т. 19, вып. 5 (в печати).
6. *Перчёнок О.В., Поздняков С.Н., Посов И.А.* Автоматизация проверки решения геометрических задач по описанию их условий на предметно-ориентированном языке // Компьютерные инструменты в образовании, 2012. № 1. С. 31–38.
7. *Препарата Ф., Шеймос М.* Вычислительная геометрия: введение. М.: Мир, 1989.
8. *Goossens M. et al.* The LaTeX Graphics companion. Pearson Education, Inc., 2008.
9. *Mertz A., Slough W.* A TikZ tutorial: Generating graphics in the spirit of TEX // TUGboat, Volume 30 (2009), № 2. P. 214–226.
10. *Seyfarth B. R.* Diag: A Drawing Preprocessor for LaTeX // TUGboat, Volume 13 (1992). № 4. P. 478–485.

11. *Tantau T.* The TikZ and PGF Packages / http://sourceforge.net/projects/pgf (дата обращения 25.02.2013), 2012.
12. http://exponenta.ru/educat/systemat/buhvalova/index3.asp (дата обращения 25.02.2013).

Abstract

The paper introduces an implementation of the geometrical specification language **DROL** originally designed to describe geometric structures and to write algorithms solving geometric problems on a plane. **DROL** can be also used to create banks of standard figures for courses, presentations and publications. The preprocessor **DROL2TeX** transforms **DROL** program into a set of commands of the graphics package **TikZ**. The result is the desired figure. We use examples to demonstrate extensions to the graphical capabilities of LaTeX.

Keywords: geometrical specification language, language **DROL**, OOP, LaTeX, graphic packages.

Бухвалова Вера Вацлавовна,
кандидат физико-математических наук, доцент кафедры исследования операций СПбГУ,
vera_cut@mail.ru,

Зациорский Артем Сергеевич,
аспирант кафедры исследования операций СПбГУ,
amartel@yandex.ru