СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность. Любое производство, связанное с изготовлением деталей из сырья сталкивалось с вопросом размещения заготовок деталей. Процесс размещения заготовок на листе называется составлением карты раскроя материала. Данный процесс зачастую влияет на эффективность работы производства, а его грамотная реализация снижает затраты производства на сырье.

Нередко получается так, что остатки от листа при первоначальном раскрое остаются не задействованы и складируются либо утилизируются, когда их еще можно использовать. Такие остатки от раскроя листового материала называют деловыми остатками. Вопрос использования деловых остатков актуален в настоящее время, поскольку его использование решает ряд задач производства по хранению, транспортировке, учету и утилизации остатков.

Трудность составления карт раскроя на деловом остатке, в отличии от листа прямоугольной формы, заключается в том, что чаще всего он имеет нестандартную форму, тем самым затрудняя размещение заготовок для раскроя.

Для того, чтоб разместить заготовки на деловом остатке, необходимо совершить полный перебор вариантов раскроя, что занимает большую часть времени, особенно если данный процесс происходит вручную. В работе производства данный процесс может повлиять в отрицательную сторону его работы.

Следовательно, автоматизация процесса рационального размещения заготовок на деловом остатке нестандартной формы положительно повлияет на работу производства, поскольку уменьшит время на создание карт раскроя, приведет к минимизации затрат на сырье и приведет к эффективному использованию уже имеющихся ресурсов. Так же эффективное использование деловых остатков благотворно повлияет на экологическую обстановку. Время разложения таких материалов как листы ДСП, стекла либо железа зачастую больше 100 лет, а его эффективное использование снизит процент загрязнения природы, тем самым снизит вред здоровью для живых организмов.

Объектом исследования является задача раскроя деловых остатков.

Предметом исследования является оптимизация составления карт раскроя деловых остатков.

Цель работы заключается в разработке программного обеспечение рационального размещения графических примитивов на деловом остатке нестандартной формы.

Задачи исследования. Для достижения поставленной цели сформированы следующие задачи:

— Выполнить анализ предметной области;

— Разработать математическую модель рационального размещения графических примитивов на деловом остатке нестандартной формы;

— Спроектировать и реализовать программное обеспечение рационального размещения графических примитивов на деловом остатке нестандартной формы на основе разработанной математической модели;

— Выполнить экономическое обоснование.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ составления карт раскроя листового материала

Любое производство, использующее листовой материал в качестве сырья для изготовления шаблонов, сталкивалось с деловыми остатками.

Деловые отходы (деловой остаток) – отходы, пригодные для использования при изготовлении изделий различного назначения (например, деловые металлоотходы по ГОСТ 16482). [1]

Другими словами деловые остатки называют остатки материала от раскроя, которые складируют и, в дальнейшем, используют для производства деталей, т.е. тоже раскраивают. Их рассматривают, как листы меньшего размера, у которых с некоторых сторон обзол уже снят.

Исследование бизнес-процесса учета делового остатка при раскрое листовых материалов рассматривается в работе [2]. Деловой остаток, как правило, в дальнейшем подвергается складированию и/или утилизации. В большинстве случаев, деловой остаток вообще не подвергается контролю [3] хотя, при грамотном его контроле и дальнейшем использовании можно минимизировать отходы производства, тем самым снизить затраты производства на материал.

Алгоритм расположения графических примитивов (далее ГО) на деловом остатке (далее ДО) применяется в составлении карт раскроя ДО с целью минимизации остатков от раскроя листового материала.

Впервые вопрос создания крат раскроя поднял П.Л.Чебышев [4] в докладе «О кройке одежды» в 1878 г. После чего тема раскроя благополучно развивалась и находила свое место в работах таких ученых как Е.С.Федоровой в работе [5], Л.В.Канторовичем и В. А. Залгаллером в работах [6] – [9], Л.В.Канторовича и В. А. Залгаллера в работе [10] и многих других.

Анализом составления карт раскроя листового материала прямоугольной формы и/или полубесконечной полосы занимали такие авторы как Ситдикова Д.В. в работе [11], А. Ю. Васин, в. Н. Задорожный в работе [12], А. Ф. Валеева, А. А. Петунин, Р. И. Файзрахманов в работе [13].

В наше время в вопросах создания карт раскроя широко применяют современные технологии. Существует огромный выбор CAD/CAM систем, специализированных в данной области. Наиболее известны программные комплексы: Aptia Solutions, Astra S-nesting, JetCam, САПР Сириус, PaneCut, Техтран, Винтех RCAM-Pro, Интех-Раскрой W/L,

NestLib (Geometric Software Solutions), Orion, NestingIntelligentSoftware, MazakSmartSystem, LantekExpert, Wrykrys, и др.

Функциональные возможности систем в сфере создания карт раскроя сравнительно схожи между собой.

Таким образом, вопросами создания карт раскроя занимались с давних времен. Большинство работ, связанных с созданием карт раскроя, рассматриваются на прямоугольной форме листа либо на полубесконечной полосе. Деловой остаток чаще рассматривают как Г-образную форму, нежели нестандартную.

1.2 Анализ моделей формирования раскроя и методов классификации

Задачи нерегулярного размещения геометрических объектов сложных форм относятся к классу задач двумерного раскроя упаковки (далее Р-У).

Общая классификация задач РУ представлена на рисунке 1. ГО, при создании карт раскроя, могут иметь различную форму.

Рисунок 1 - Общая классификация задач раскроя упаковки (РУ)

В Р-У фигурная укладка может быть регулярной или не регулярной.

Суть регулярного размещения ГО заключается в том, что объекты располагаются относительно друг друга на определенную константу и одинаково ориентированы относительно листа. [30] Вопрос регулярного размещения ГО в достаточной мере изучен. Его можно решить, используя точные методы, такие как аппрокация и декомпозиция. Подробное описание точных методов регулярного размещения ГО рассматривается в работе [14].

Для нерегулярного размещения ГО точные методы решения данной задачи будут недостаточны и не приведут к ожидаемому оптимуму. Основные подходы к нерегулярному размещению ГО сложной формы представлены на рисунке 2.

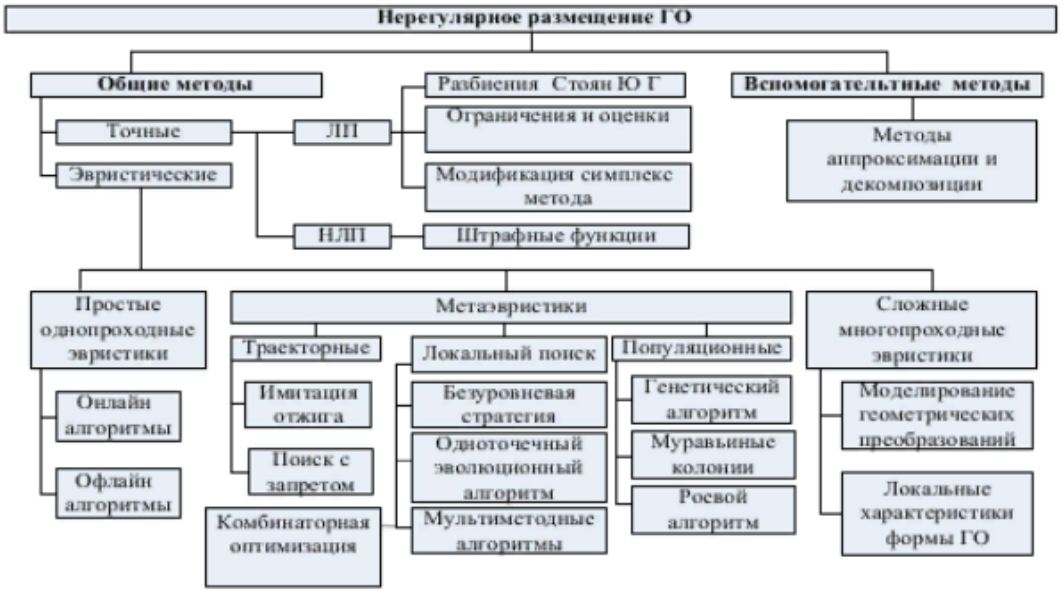


Рисунок 2 - Основные подходы к нерегулярному размещению ГО сложной формы

Эвристические методы, как видно, часто используемые в вопросе нерегулярного размещения ГО.

Из них, по популярности, можно выделить следующие:

— «Генетический алгоритм» (ga). Суть алгоритма описывается в работах [13], [15];

— Метод «муравьиной колонии» (ac). Суть алгоритма описывается в работе[15];

— Метод «моделирование отжига» (sa). Суть алгоритма описывается в работах [13], [15];

— Метод «поиск с запретами» (ts). Суть алгоритма описывается в работах [13], [15], [16];

— Метод последовательного уточнения оценок (svc).Суть алгоритма описывается в работе [13];

— Алгоритм метода комбинаторной оптимизации (grasp). Суть алгоритма описывается в работе [13];

— Гибридный алгоритм оптимального размещения лекал, совмещающий в себе идеи муравьиных колоний и «жадной» стратегии. Суть алгоритма описывается в работе [13];

— Алгоритм анализа локальных характеристик формы геометрических объектов. Суть алгоритма описывается в работе [20];

— Методы моделирования геометрических преобразований. Суть алгоритма описывается в работе [17];

— Алгоритмы пороговой допустимости (ta). Суть алгоритма описывается в работе [17];

— Алгоритм перехода от рекорда к рекорду (rtr). Суть алгоритма описывается в работе [17];

— Аппроксимация методом касательных. Суть алгоритма описывается в работе [18];

— Алгоритм группировки геометрических объектов с использованием локальных характеристик формы. Суть алгоритма описывается в работе [19].

Более детальный обзор алгоритмов и методов нерегулярного фигурного раскроя приведен в работе [21].

В работе [22] описываются алгоритмы фигурного раскроя такие как: точные алгоритмы, геометрические алгоритмы, пиксельно-растровый методы и методы с применением тригонометрических функций, так же уделяется внимание эвристическим алгоритмам. Описаны методы линейного программирования, различные модификации метода ветвей и границ, разобраны алгоритмы локального поиска, затронут «жадный» алгоритм.

Таким образом, определили, что в данной работе рассматривается двумерный раскрой упаковки на фигурные ГО способом их нерегулярного размещения. Алгоритмов создания карт раскроя достаточно много.

Вывод: рассмотрена общая история развития вопросов создания карт раскроя. Просмотрены работы раскроя материала прямоугольной формы и полубесконечной полосы. Деловой остаток чаще рассматривают как Г-образную форму, нежели нестандартную. Определена классификация создания карт раскроя, рассматриваемая в данной работе.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

2.1 Математическая постановка задачи составления карт раскроя ДО

В данной работе рассматривается двумерный раскрой упаковки на фигурные ГО способом их нерегулярного размещения.

Используя ДО нестандартной формы необходимо составить k различных вариантов карт раскроя, используя n-видов ГО i-ого (i=1..n) типа, таким образом, чтоб площадь остатка ДО сводился к минимуму.

ДО имеет площадь S ($мм^2$) и m точек перегиба контура с координатами, представленными в виде двумерного массива $C=\{C_{i'j'}, i'=1..m, j'=1..2\}$. $C[i',j']$ – координата i'-ой точки перегиба контура ДО по оси j'.

ГО характеризуются площадью $S=\{s_i, i=1..n\}$ измеряемой в $мм^2$, количеством точек перегиба контура $T=\{t_i, i=1..n\}$ с координатами, представленными в виде двумерного массива $C_i=\{C_{jj'}, j=1..t_i, j'=1..2\}$. $C_i=[j,j']$ – координата j-ой точки перегиба контура фигуры i-ого вида по оси j'.

Математическая постановка задачи составления карт раскроя ДО состоит в определении минимального значения целевой функции, определяемой по формуле (1).

$$F=S-\sum_{i=1}^{n} S_i \cdot x_i \rightarrow \min \quad (1)$$

где x_i – это количество ГО i-ого типа, участвующего в раскрое.

В данной работе используются 3 метода формирования карт раскроя ДО нестандартной формы:

1) Метод оптимизации «рюкзака»;
2) Метод комбинаторики с применением жадного алгоритма;
3) Метод комбинаторики с применением полного перебора.

Таким образом, необходимо составить различные варианты карт раскроя тремя методами: методом оптимизации «рюкзака», методом комбинаторики с применением жадного алгоритма, методом комбинаторики с применением полного перебора.

2.2 Алгоритм сохранения координат ГО в зависимости от типа

Сохранение первоначальных данных ГО и их координат точек перегиба контура производится способом соблюдения правил геометрии. Общая схема алгоритма ввода данных ГО представлена на рисунке 3.

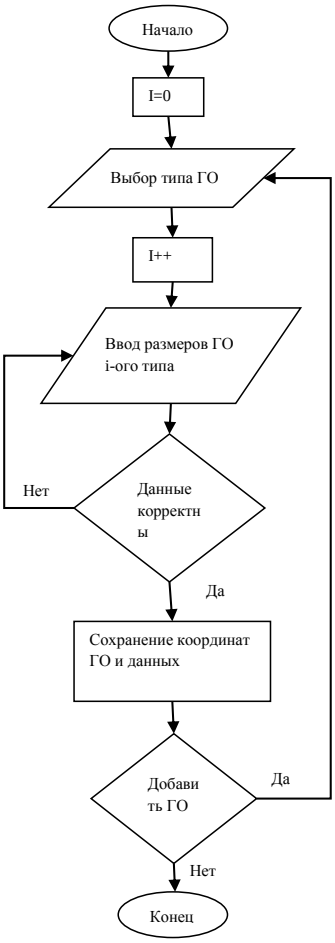


Рисунок 3 - Общая схема алгоритма ввода данных ГО

2.2.1 Тип «Треугольник»

Для сохранения данных треугольника достаточно знать размер трех его сторон: a, b, c (мм), при соблюдении условий (2). [25]

$$\begin{cases} a, b, c>0 \\ \left[\begin{array}{l} a+b>c \\ a+c>b \\ b+c>a \end{array}\right. \end{cases} \quad (2)$$

Другими словами, все стороны треугольника должны быть больше нуля, и большая его сторона должна быть меньше суммы двух оставшихся.

При соблюдении условий (2) площадь треугольника находится по формуле (3).

$$S_{\Delta}=\frac{a\cdot h}{2} \tag{3}$$

где h–это высота, опущенная к стороне a (см. рис. 4).

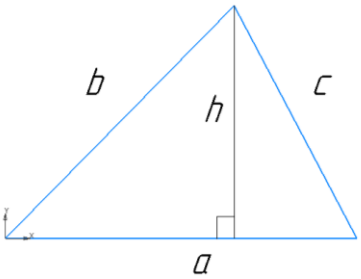


Рисунок 4 - Треугольник

По правилам прямоугольного треугольника, высоту, опущенную к стороне a, находится по формуле (4).

$$\begin{cases} h=\sqrt[2]{b^2-x^2} \\ x=\frac{a^2+b^2-c^2}{2\cdot a} \end{cases} \tag{4}$$

где x–расстояние от начала стороны a до h.

Тогда координаты треугольника в матрицу C_i сохраняются следующим образом (см. рис. 5) находятся по формуле (5).

$$C_i=\begin{pmatrix} 0 & 0 \\ x & h \\ a & 0 \end{pmatrix} \text{прит}_i=3 \tag{5}$$

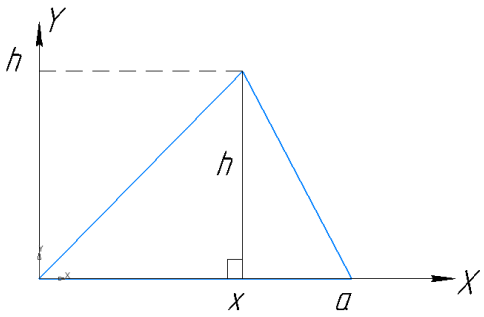


Рисунок 5 - Определение координат треугольника

2.2.2 Тип «Квадрат»

Для сохранения данных квадрата достаточно знать размер его сторон: a, b (мм), при a и b больше нуля. [15]

Площадь квадрата находится по формуле (6).

$$S=a\cdot b \tag{6}$$

Координаты квадрата в матрицу C_i сохраняются следующим образом (см. рис. 6).

$$C_i=\begin{pmatrix}0 & 0\\ 0 & b\\ a & b\\ a & 0\end{pmatrix} \text{прит}_i=4 \tag{7}$$

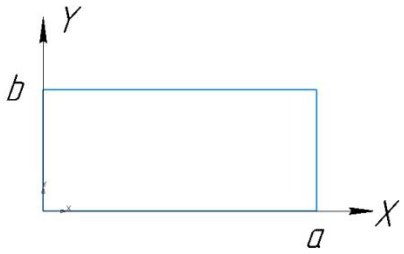


Рисунок 6 - Квадрат

2.2.3 Тип «Ромб»

Для сохранения данных ромба достаточно знать размер его сторонуа (мм) и большего его угла β° (см. рис 7). [25] При этом, необходимо соблюдать условия (8).

$$\begin{cases} a>0 \\ 90<\beta°<180 \end{cases} \quad (8)$$

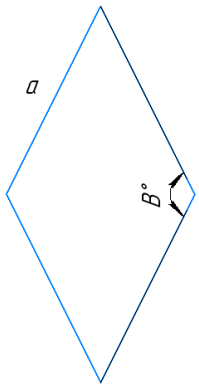


Рисунок 7 - Ромб

Введем дополнительные переменные a_1 и b_1 (см. рис 8), тогда площадь ромба находится по формуле (9).

$$S=a\cdot b_1 \quad (9)$$

где b_1 – это перпендикуляр, опущенный к стороне (см. рис. 8).

По правилам тригонометрии размеры a_1 и b_1 находятся по формуле (10).

$$\begin{aligned} b_1&=\sin(180-\beta)\cdot a \\ a_1&=\cos(180-\beta)\cdot a \end{aligned} \quad (10)$$

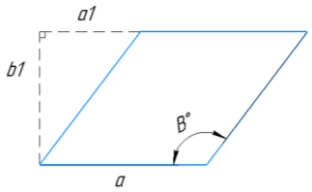


Рисунок 8 - Перпендикуляры к ромбу

Координаты ромба в матрицу C_i сохраняются следующим образом (см. рис. 9).

$$C_i=\begin{pmatrix} 0 & 0 \\ a_1 & b_1 \\ a+a_1 & b_1 \\ a & 0 \end{pmatrix} \text{прит}_i=4 \quad (11)$$

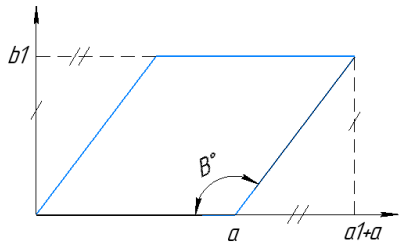


Рисунок 9 - Определение координат ромба

2.2.4 Тип «Трапеция»

Для сохранения данных трапеции достаточно знать размер его сторон: a, b, c (мм) и меньшего его угла β° при основании(см. рис 10). [25] При этом, необходимо соблюдать условия (12).

$$\begin{cases} a, b, c>0 \\ a>b \\ 0<\beta°<90 \end{cases} \quad (12)$$

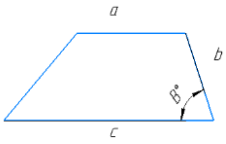


Рисунок 10 – Трапеция

Площадь трапеции находится по формуле (13).

$$S=\frac{(a+b)\cdot c\cdot \sin(\beta)}{2} \quad (13)$$

Для поиска координат введем дополнительные переменные: a_1, b_1 (см. рис. 11). Тогда Координаты трапеции в матрицу C_i сохраняются в виде (11) при следующих значениях переменных.

$$\begin{aligned} a_1&=a-c\cdot \cos(\beta)-b \\ b_1&=\sin(\beta)\cdot c \end{aligned} \quad (14)$$

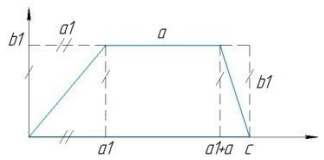


Рисунок 11 - Определение координат трапеции

2.2.5 Тип «Произвольный»

Координаты произвольного ГО сохраняются в непосредственном виде.

Площадь ГО при t_i=3 находится по формуле (3) размер сторон при этом находятся по формуле (30) как отрезок между вершинами треугольника (см. рис. 12).

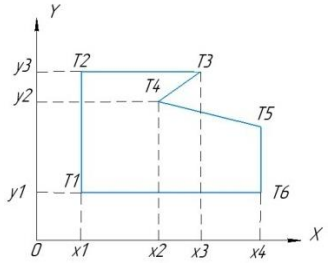


Рисунок 12 - Пример произвольной фигуры

Площадь ГО при t_i=3находится по формуле площади гаусса (15).

Формула площади Гаусса(формула шнурованияилиалгоритм шнурования)— формула определенияплощадипростогомногоугольника, вершины которого заданыдекартовыми координатамина плоскости. [26]

$$S=\frac{|a-b|}{2} \tag{15}$$

где a, b– переменные накапливания результата, находятся по формуле (16).

$$\begin{aligned} a&=a+C_{i'1}\cdot C_{i'+12}, \text{ где } i'=1..(t_i-1) \\ b&=b+C_{i'2}\cdot C_{i'+11}, \text{ где } i'=1..(t_i-1) \end{aligned} \tag{16}$$

Таким образом, сохранение первоначальных данных ГО и их координат точек перегиба контура производится способом соблюдения правил геометрии. Непосредственно координаты вводятся для ГО типа «произвольный». Для других типов ГО, относящихся к стандартным фигурам первоначальными данными являются размеры сторон фигуры в мм либо один из углов фигуры.

2.3 Алгоритм максимального расположения ГОi-ого вида на ДО

Если представить ДО в виде матрицы DO[y,x] размером S´xD´, где S´ и D´ рассчитываются по формуле (17), имеющей значения 0 или 1, и фигуру i-ого вида в виде матрицы Sh размером SxDтак же имеющей значения 0 или 1, то с помощью наложения матрицы Sh на матрицу DO в точку DO[y,x] можно определить: войдет ли ГОi-ого вида на ДО и определить какие она имеет точные координаты местоположения.

$$\begin{matrix} S'=\max(C_{i'2}) \\ D'=\max(C_{i'1}) \end{matrix}, i' = 1..j \quad (17)$$

2.3.1 Представление фигуры в виде матрицы

Рассмотрим процесс представления фигуры i-ого вида (далее F_i) (см. рис. 13) в виде матрицы Sh_{SxD}.

$Sh_{S\,xD}=(Sh_{y_0x_0})$ – матрица F_i, где строки соответствуют оси Y, столбцы соответствуют оси X.

Размерность матрицы фигуры рассматривается как, длинна и ширина фигуры относительно осей координат.

Ширина (S) F_i – это разница между максимальной и минимальной координатами фигуры относительно оси Y. Длинна (D) F_i – это разница между максимальной и минимальной координатами фигуры относительно оси X.

$$\begin{matrix} S=\max(C_{i_02}) \\ D=\max(C_{i_01}) \end{matrix}, i_0 = 1..ti. \quad (18)$$

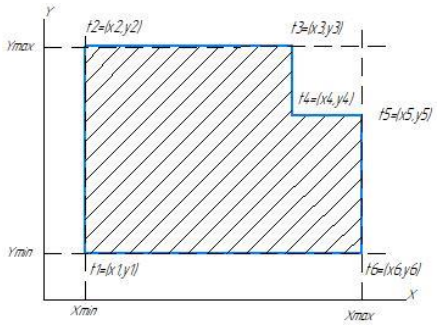


Рисунок 0-13 – Пример фигуры F_i

Для того, чтоб заполнить матрицу $Sh_{S\,xD}[y_0,x_0]$ значениями 1 или 0 необходимо методом перебора строк и столбцов матрицы определить место нахождения точки A=(x0,y0) относительно F_i, имеющей t_i точек перегиба контура. Если точка A=(x0,y0) лежит на плоскости фигуры, то матрица в данном положении будет иметь значение 1, если A=(x0,y0) не лежит на плоскости фигуры, то матрица в данном положении будет иметь значение 0.

Если $A=(x0,y0)\in F_i$ то Sh[y0,x0]=1, иначе Sh[y0,x0]=0.

Различное множество алгоритмов определения положения точки относительно многоугольника рассматриваются в работе [20]:

— Алгоритм для выпуклых многоугольников, в частности для треугольников;

— Тригонометрический алгоритм;

— Триангуляционный алгоритм;

— Определение по площади для выпуклых многоугольников, в частности для треугольников;

— Лучевой алгоритм;

— Инцидентный лучевой алгоритм;

— Инцидентный лучевой алгоритм для многогранников;

— ER – модель.

Более детальное разъяснение лучевого метода нахождения точки относительно плоскости рассматривается в работе [27].

Точка A=(x0,y0) может находиться в одном из трех положений:

— Внутри плоскости F_i (точка A1 рисунок 14);

— На границах контура F_i (точка A2 рисунок 14);

— Вне плоскости F_i (точка A3 рисунок 14).

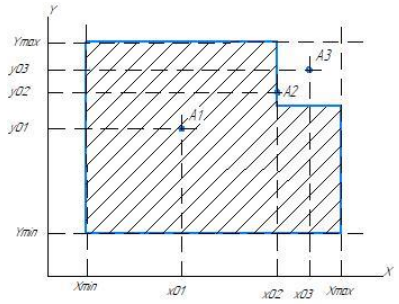


Рисунок 14 - Варианты положения точки A

Определение местоположения точки A=(x0,y0) условно делится на этапы: проверка нахождения точки на осях F_i; проверка нахождения точки либо внутри либо снаружи плоскости F_i.

Если F_i имеет t_i точек перегиба контура, то можно сказать, что фигура описывается t_i уравнениями вида (19).

$$\begin{cases} A\cdot x+B\cdot y+C=0 \\ A=y_2-y_1 \\ B=x_1-x_2 \\ C=y_1\cdot(x_2-x_1)-x_1\cdot(y_2-y_1) \end{cases} \quad (19)$$

где— произвольные постоянные, причём постоянныене равны нулю одновременно;

(x_1,y_1) – координата первой точки прямой;

(x_2,y_2) – координата второй точки прямой [25].

В случае, если прямая Lпроходит через точки $Z=(C_{j\,1},C_{j\,2})$ и $Z'=(C_{j+1\ 1},C_{j+1\ 2})$ F_i, то уравнение прямой находится по формуле (20).

$$(C_{j+1\ 2}-C_{j\,2})\cdot X+(C_{j\,1}-C_{j+1\ 1})\cdot Y+(C_{j\,2}\cdot(C_{j+1\ 1}-C_{j\,1})-C_{j\,1}\cdot(C_{j+1\ 2}-C_{j\,2}))=0 \quad (20)$$

Точка $A=(x_0,y_0)$ принадлежит прямой L если уравнение (20) при $x=x_0$, $y=y_0$ будет равняться 0.

При определении принадлежности точки A прямой L необходимо учитывать, что точка может принадлежать прямой, но находится вне границах отрезка $\overline{ZZ'}$, где $Z=(C_{j\,1},C_{j\,2})$, $Z'=(C_{j+1\ 1},C_{j+1\ 2})$ (см. рис. 15). Следовательно, координата (x_0,y_0) так же должна удовлетворять условиям (21).

$$\begin{cases} C_{j\,1}\le x_0\le C_{j+1\ 1} \\ C_{j\,2}\le y_0\le C_{j+1\,2} \end{cases} \quad (21)$$

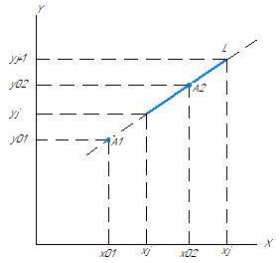


Рисунок 15 – Определение положение точки A относительно прямой

Для того чтоб проверить точку A=(x0,y0) на нахождение относительно прямых, описывающих F_i, необходимо способом перебора прямых в количестве t_i, составить их уравнения в виде (20), найти решение уравнения при $x=x_0$, $y=y_0$. В случае если хотя бы 1 уравнение равняется нулю, то точка лежит на данной прямой. Далее необходимо проверить точку на удовлетворение условий (21) и если рассматриваемая точка удовлетворяет условиям, то можно сказать, что точка принадлежит фигуре, матрица Sh[y0,x0] принимает значение 1.

$$
\begin{gathered}
Sh[y_o x_0]=1 \\
\left[\begin{array}{c}
\left\{\begin{array}{c}\left(C_{2\,2}-C_{1\,2}\right)\cdot X_0+\left(C_{1\,1}-C_{2\,1}\right)\cdot Y_0+\left(C_{12}\cdot\left(C_{2\,1}-C_{1\,1}\right)-C_{1\,1}\cdot\left(C_{2\,2}-C_{1\,2}\right)\right)=0 \\ C_{1\,1}\le x_0\le C_{2\,1} \\ C_{1\,2}\le y_0\le C_{2\,2}\end{array}\right. \\
\left\{\begin{array}{c}\left(C_{3\,2}-C_{2\,2}\right)\cdot X_0+\left(C_{2\,1}-C_{3\,1}\right)\cdot Y_0+\left(C_{2\,2}\cdot\left(C_{3\,1}-C_{2\,1}\right)-C_{2\,1}\cdot\left(C_{3\,2}-C_{2\,2}\right)\right)=0 \\ C_{2\,1}\le x_0\le C_{3\,1} \\ C_{2\,2}\le y_0\le C_{3\,2}\end{array}\right. \\
\cdots \\
\left\{\begin{array}{c}\left(C_{1\,2}-C_{t_i\,2}\right)\cdot X_0+\left(C_{t_i\,1}-C_{1\,1}\right)\cdot Y_0+\left(C_{t_i\,2}\cdot\left(C_{1\,1}-C_{t_i\,1}\right)-C_{t_i\,1}\cdot\left(C_{1\,2}-C_{t_i\,2}\right)\right)=0 \\ C_{1\,1}\le x_0\le C_{t_i\,1} \\ C_{1\,2}\le y_0\le C_{t_i\,2}\end{array}\right.
\end{array}\right.
\end{gathered}
\tag{22}
$$

Для того, чтоб проверить нахождение точки A=(x0,y0) относительно плоскости F_i не зависимо от контуров фигуры, необходимо построить дополнительную прямую E, проходящую через точку (x_0,y_0) и параллельную оси Ox. Если прямая E имеет четное количество точек пересечения с гранями F_i, где точка пересечения A′ имеет координаты (x′,y′), удовлетворяющие условиям $y_0=y'$, $x_0>x'$, то точка $A=(x_0,y_0)$ лежит снаружи F_i. Если точек пересечения прямой E с гранями F_i, удовлетворяющих ранее описанным условиям, имеется нечетное количество, то точка $A=(x_0,y_0)$ лежит на плоскости F_i (см. рис. 16).

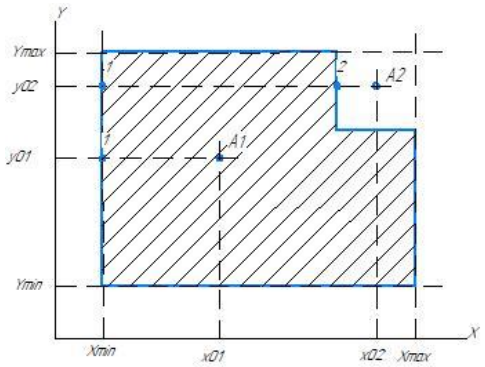


Рисунок 16 - Варианты положения точки A

Прямая E, проходящая через точку A=(x0,y0) и параллельная оси Ox по уравнению (19 – 20) при значении коэффициентов находится по формуле (23).

$$\begin{cases} -x_0 \cdot y + y_0 \cdot x_0 = 0 \\ A = y_2 - y_1 = y_0 - y_0 = 0 \\ B = x_1 - x_2 = 0 - x_0 = -x_0 \\ C = y_1 \cdot (x_2 - x_1) - x_1 \cdot (y_2 - y_1) = y_0 \cdot x_0 \end{cases} \quad (23)$$

Если прямая L проходит через точки $Z=(C_{j\,1},C_{j\,2})$ и $Z'=(C_{j+1\;\,1},C_{j+1\;\,2})$ F_i, то точку пересечения прямой L и E можно найти, решив систему уравнений (20) и (23).

$$\begin{cases} -x_0 \cdot y' + y_0 \cdot x_0 = 0 \\ (C_{j+1\;\,2} - C_{j\,2}) \cdot x' + (C_{j\,1} - C_{j+1\;\,1}) \cdot y' + (C_{j\,2} \cdot (C_{j+1\;\,1} - C_{j\,1}) - C_{j\,1} \cdot (C_{j+1\;\,2} - C_{j\,2})) = 0 \end{cases} \quad (24)$$

Решив систему любым известным методом (Гаусса, Крамера и др.) найдем точку пересечения A′=(x′,y′). При этом, аналогично предыдущим вычислениям, учитывается то, что прямая Lограничивается двумя точками, следовательно точка A′ должна удовлетворять условиям (25).

$$\begin{cases} C_{j\,1} \le x' \le C_{j+1\;\,1} \\ C_{j\,2} \le y' \le C_{j+1\,2} \end{cases} \quad (25)$$

Так же координата точки A′=(x′,y′) по оси X должна быть меньше рассматриваемой точки A=(x0,y0).

$$\begin{cases} x' < x_0 \\ y' = y_0 \end{cases} \quad (26)$$

Для того чтоб проверить точку A=(x0,y0) на нахождение относительно F_i, необходимо задать переменную p целочисленного типа, которая будет накапливать количество точек пересечения, удовлетворяющих условиям (25 – 26). Составить уравнение прямой проходящей через точку A=(x0,y0) и параллельной оси X в виде (23), способом перебора прямых в количестве t_i, составить их уравнения в виде (20), решив систему (24) найти точку пересечения A′=(x′,y′). Далее проверить точку A′=(x′,y′) на

удовлетворение условий (25) и (26), в случае если A′=(x′,y′) удовлетворяет условиям, то p увеличивается.

Если p четное, то точка A′=(x′,y′) находится вне F_i, если нечетное, то в плоскости F_i.

Поскольку при переборе прямых, описывающих F_i, точки пересечения прямой E с контуром F_i могут совпадать, то в учет принимается только один экземпляр точки. Если прямая E пересекает прямую L параллельную оси Ox (A=0, B<>0) то рассматриваются лишь крайние точки отрезка и число p увеличивается на 2 (см. рис. 17).

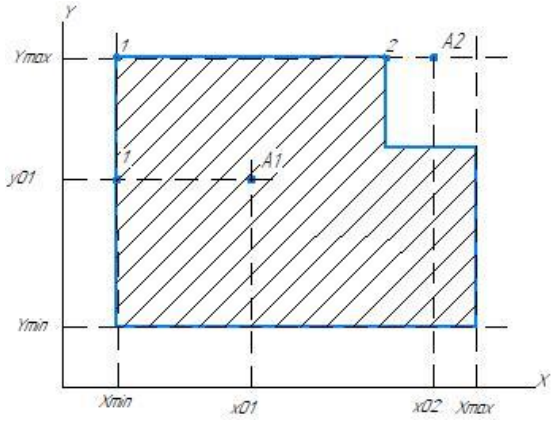


Рисунок 17 – Варианты положения точки A

$$
\begin{cases}
\text{если } \begin{cases} C_{2\,2}-C_{12}<>0 \\ C_{1\,1}-C_{2\,1}<>0 \end{cases} \text{ то } \begin{cases} -x_0\cdot y'+y_0\cdot x_0=0 \\ (C_{2\,2}-C_{12})\cdot x'+(C_{1\,1}-C_{2\,1})\cdot y'+(C_{1\,2}\cdot(C_{2\,1}-C_{1\,1})-C_{1\,1}\cdot(C_{2\,2}-C_{1\,2}))=0 \\ C_{1\,1}\le x'\le C_{2\,1} \\ C_{2\,2}\le y'\le C_{2\,2} \\ x'<x_0 \\ y'=y_0 \end{cases} \text{ p=p+1,} \\
\text{иначе если } \begin{cases} C_{2\,2}-C_{12}=0 \\ C_{1\,1}-C_{2\,1}<>0 \end{cases} \text{ то p=p+2} \\
\text{если } \begin{cases} C_{3\,2}-C_{2\,2}<>0 \\ C_{2\,1}-C_{3\,1}<>0 \end{cases} \text{ то } \begin{cases} -x_0\cdot y'+y_0\cdot x_0=0 \\ (C_{3\,2}-C_{2\,2})\cdot x'+(C_{2\,1}-C_{3\,1})\cdot y'+(C_{2\,2}\cdot(C_{3\,1}-C_{2\,1})-C_{2\,1}\cdot(C_{3\,2}-C_{2\,2}))=0 \\ C_{2\,1}\le x'\le C_{3\,1} \\ C_{2\,2}\le y'\le C_{3\,2} \\ x'<x_0 \\ y'=y_0 \end{cases} \text{ p=p+1,} \\
\text{иначе если } \begin{cases} C_{3\,2}-C_{2\,2}=0 \\ C_{2\,1}-C_{3\,1}<>0 \end{cases} \text{ то p=p+2} \\
\dots \\
\text{если } \begin{cases} C_{1\,2}-C_{j\,2}<>0 \\ C_{j\,1}-C_{1\,1}<>0 \end{cases} \text{ то } \begin{cases} -x_0\cdot y'+y_0\cdot x_0=0 \\ (C_{1\,2}-C_{j\,2})\cdot x'+(C_{j\,1}-C_{1\,1})\cdot y'+(C_{j\,2}\cdot(C_{1\,1}-C_{j\,1})-C_{j\,1}\cdot(C_{1\,2}-C_{j\,2}))=0 \\ C_{1\,1}\le x'\le C_{j\,1} \\ C_{1\,2}\le y'\le C_{j\,2} \\ x'<x_0 \\ y'=y_0 \end{cases} \text{ p=p+1,} \\
\text{иначе если } \begin{cases} C_{1\,2}-C_{j\,2}=0 \\ C_{j\,1}-C_{1\,1}<>0 \end{cases} \text{ то p=p+2}
\end{cases} \tag{27}
$$

Если p-четное, то $Sh[y_0,x_0]=0$, иначе $Sh[y_0,x_0]=1$.

Пример итоговой матрицы фигуры представлен на рисунке 18.

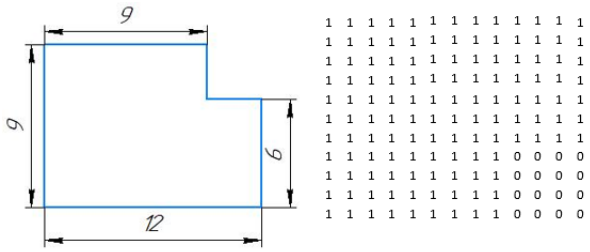


Рисунок 18 - Пример работы составления матрицы фигуры

Таким образом, алгоритм представления фигуры в виде матрицы сводится к определении точек фигуры их местоположения. Если точка лежит на плоскости фигуры, то матрица в данном положении будет иметь значение 1, если не лежит на плоскости фигуры, то матрица в данном положении будет иметь значение 0.

Блок схема алгоритма представления ГО i-ого типа в виде матрицы представлен в приложении А.

2.3.2 Наложение матрицы ГО на матрицу ДО

Наложение матрицы Sh ГО на матрицу делового остатка DO происходит путем перебора матрицы ДО с левого нижнего угла вдоль горизонтальной прямой с шагом 1 мм. Если DO[y,x]=1, то в точке A=(x,y) есть вариант того, что шаблон войдет. DO[y,x] рассматривается как константа (базовая точка наложения матрицы F_i). Из рассматриваемой точки делового остатка DO[y,x], при DO[y,x]=1, пытаемся вместить максимальное количество F_i путем поворота F_i из начального положения на градус α (см. рис. 19). В случае, если F_i входит на ДО и данное положение «свободно», то положение F_i сохраняется путем изменения матрицы делового остатка и сохранением координат. Таким образом, просматривается весь деловой остаток и вписываются максимальное количество шаблонов.

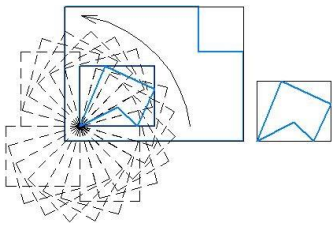


Рисунок 19 - Процесс наложения матрицы на ДО в точку A=(x,y)

F_i имеет t_i точек перегиба контура (j={1.. t_i}), с координатами, представленными в виде двумерного массива C_i={$C_{jj'}$, j=1.. t_i, j′=1..2}. C_i[j,j′] – координата i_0-ой точки перегиба контура фигуры i-ого вида по оси j′.

Координаты F_i вида j=(C_i[j,1],C_i[j,2]) считаются стандартными, угол α при этом равен 0 (α=0).

Новые координаты фигуры, при повороте на α градусов сохраняются в виде (28). [25]

$$\begin{aligned} C_i[j,1] &= C_i[j,1]\cdot \cos\left(\frac{\alpha\cdot 3.14}{180}\right) - C_i[j,2]\cdot \sin\left(\frac{\alpha\cdot 3.14}{180}\right) \\ C_i[j,2] &= C_i[j,2]\cdot \cos\left(\frac{\alpha\cdot 3.14}{180}\right) - C_i[j,1]\cdot \sin\left(\frac{\alpha\cdot 3.14}{180}\right) \end{aligned} \qquad (28)$$

Для проверки вместимости F_i на ДО в DO[y,x] при повороте на α градусов необходимо построить матрицу Sh_{SxD} по ранее разработанному алгоритму относительно новых координат, найденных по уравнению (28). При этом, необходимо учитывать то, что при наложении матрицы координаты шаблона увеличиваются на x и y соответственно.

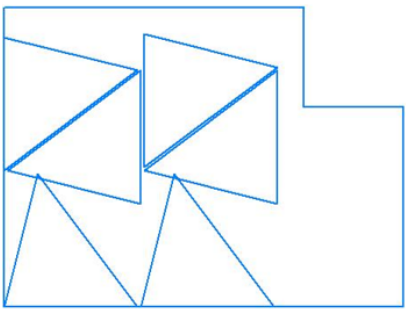
$$j=(C_i[j,1]+x,C_i[j,2]+y) \quad (29)$$

При DO[y,x]=1 перебираем матрицу Sh_{SxD} [y0,x0] слева направо сверху вниз с шагом 1. Если Sh[y0,x0]=1, то смотрится значение DO[y+y0,x+x0]. При DO[y+y0,x+x0]=1 в данное положение на данный момент времени фигура входит, в случае если Sh[y0,x0]=1, а DO[y+y0,x+x0]=0, то фигура не входит и проверка прерывается. Если после проверки все 1 из матрицы шаблона Sh[y0,x0] соответствуют 1 матрицы ДО DO[y+y0,x+x0], то фигура в данное положение входит.

При сохранении положения F_i на ДО координаты F_i увеличиваются на x и y по уравнению (29) и матрица DO в месте наложения F_i обнуляется.

Если Sh[y0,x0]=1 то DO[y+y0,x+x0]=0

Вмещение треугольника на ДО способом наложения матриц приводится на рисунке 20.



0 0 0 0 0 0 0 0 0 1 1 1 1
1 0 0 0 1 0 0 0 1 1 1 1 1
1 0 0 1 1 0 0 1 1 1 1 1 1
1 0 1 1 1 0 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 0 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 0 0 0 0

Рисунок 20 - Пример раскроя ДО на шаблоны, пример отображения матрицы ДО

Схема максимального наложения ГО i-ого типа на ДО нестандартной формы представлена на рисунке 21.

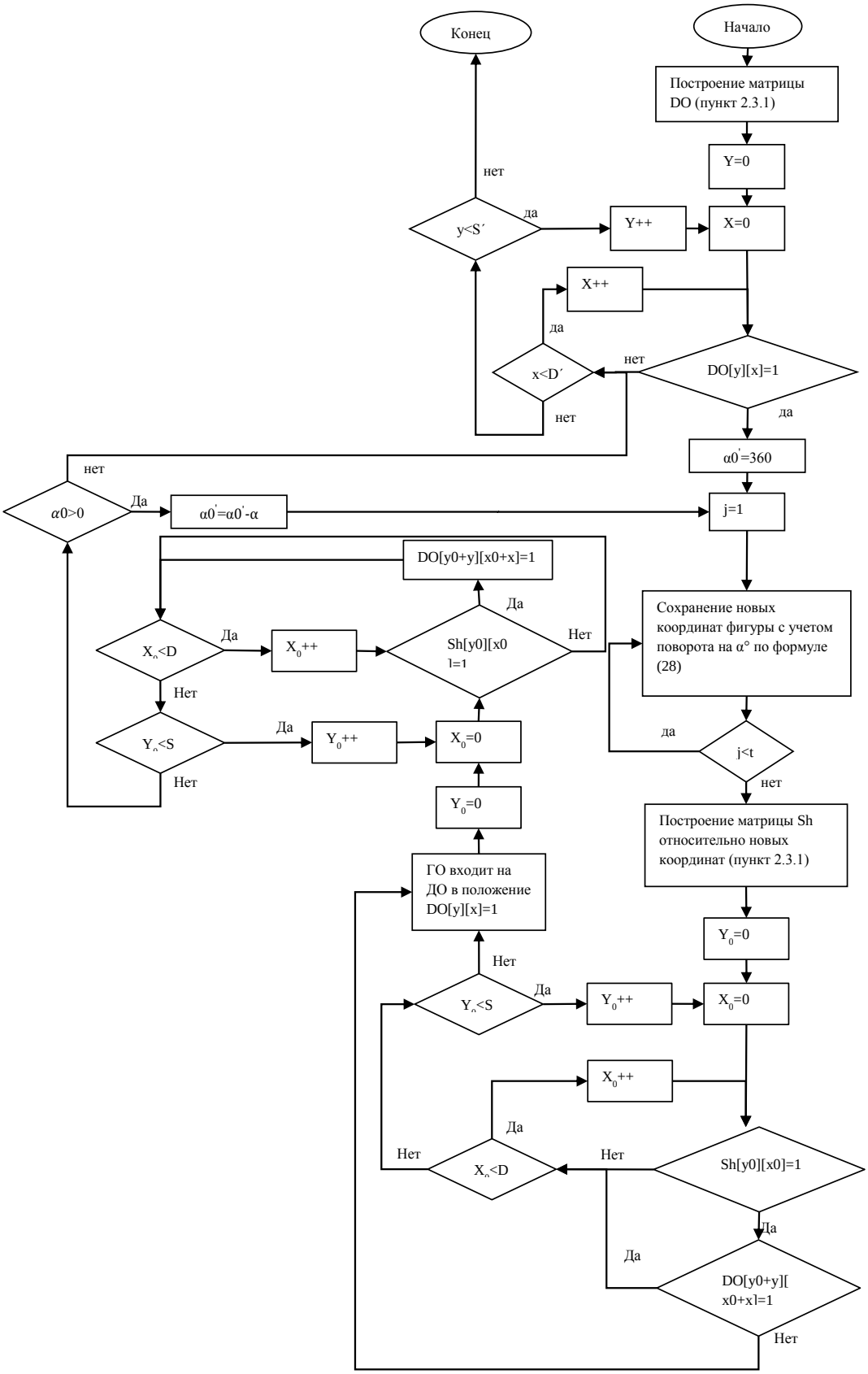


Рисунок 21 - Схема максимального наложения ГО i-ого типа на ДО нестандартной формы

Данный метод наложения фигуры на деловой остаток в виде наложения матриц основан на «жадной» стратегии, которая подразумевает, что найденное положение фигуры, удовлетворяющее условиям вхождения, на каждом этапе работы алгоритма является оптимальным, допуская, что конечное решение также окажется оптимальным.

Таким образом, ГО размещается на ДО методом наложения его матрицы на матрицу ДО с допустимым поворотом ГО на α°. Таким образом можно ответить на вопрос вхождения ГО и определить какие она имеет точные координаты местоположения.

2.4 Анализ эффективности алгоритма

Из точки A=(x,y) делового остатка можно рассмотреть максимум 360 положений F_i с углом поворота на α=1°.

Число всех возможных вариантов вмещения фигуры F_i на деловой остаток в точку A=(x,y) с углом поворота на α градусов [k] находится делением полного спектра рассматриваемого угла поворота (360°) на α°.

Максимальный угол поворота фигуры F_i при достижении оптимального результата расположения фигуры можно узнать путем нахождения угла β треугольника ABC(см. рис. 22).

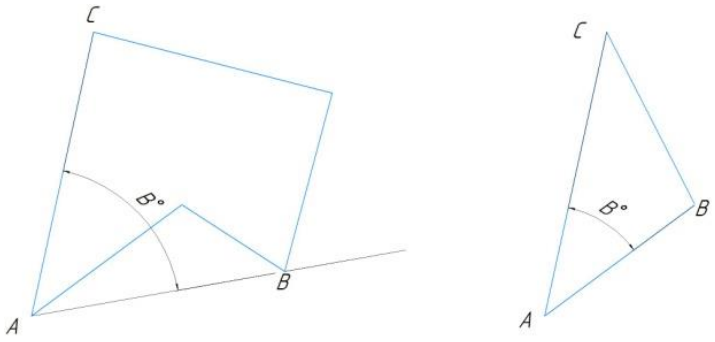


Рисунок 22 - Выбор угла β треугольника ABC

При этом, максимальное количество фигур F_i в точке A=(x,y) с углом поворота на β° [m] находится делением полного спектра рассматриваемого угла поворота (360°) на β°, где m – максимальное число благоприятных исходов вмещения фигуры F_i на деловой остаток в точкуA=(x,y) с углом поворота β° (см. рис. 23).

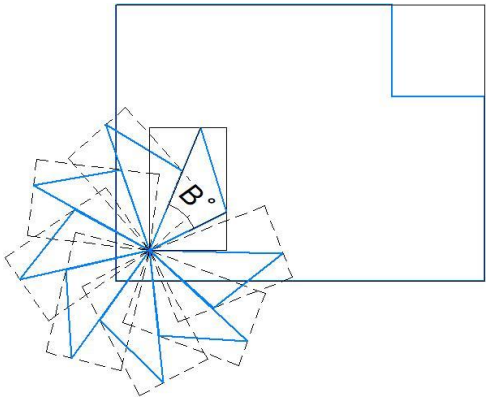


Рисунок 23 - максимальное число благоприятных исходов вмещения фигуры Fi на деловой остаток в точку A=(x,y) с углом поворота β°

K и m могут уменьшиться при учете площади делового остатка в рассматриваемой области полного спектра угла поворота. S – площадь рассматриваемого участка ДО, благоприятная для размещения F_i (рис. 24).

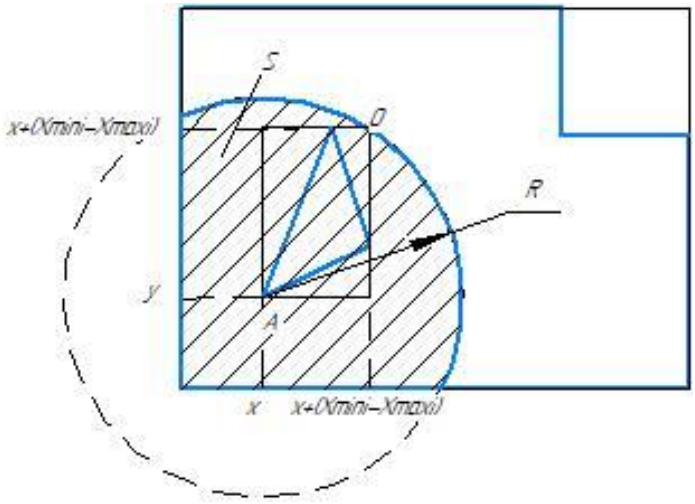


Рисунок 24 – Отображение поверхности рассматриваемого участка ДО, благоприятная для размещения F_i

R – это длинна отрезка AO, проходящего через точки A=(x,y) и O=(x+max$(C_{i_0 1})$-min$(C_{i_0 1})$,max$(C_{i_0 2})$) где $i_0=1..t_i$.

$$R=\sqrt[2]{((x+\max(C_{i_0 1})-\min(C_{i_0 1}))-x)^2+((y+\max(C_{i_0 2})-\min(C_{i_0 2}))-y)^2}=$$

$$=\sqrt[2]{(\max(C_{i_0 1})-\min(C_{i_0 1}))^2+(\max(C_{i_0 2})-\min(C_{i_0 2}))^2} \quad (30)$$

Таким образом, число всех возможных вариантов вмещения фигуры Fiна деловой остаток в точкуA=(x,y) с углом поворота на α градусов [k] и максимальное количество фигур F_i в точке A=(x,y) с углом поворота на β° [m] при учете площади, допустимой для размещения фигуры F_i находятся по формулам (31)-(32).

$$m=\frac{S}{\frac{360}{\beta}} \tag{31}$$

$$k=\frac{S}{\frac{360}{\alpha}} \tag{32}$$

Для рационального рассмотрения вариантов расположения фигуры F_i в точке A=(x,y) число всех возможных вариантов вмещения фигуры F_i на деловой остаток в точку A=(x,y) с углом поворота на α° должен быть не менее максимального количества фигур F_i в точке A=(x,y) с углом поворота на β° при учете площади, допустимой для их размещения.

$$k\geq m \tag{33}$$

Следовательно, рациональность достигается при α≤β.

При соблюдении условия (33) вероятность того, что фигура F_i вместится на деловой остаток в точку A=(x,y) равна отношением m на k.

$$P=\frac{m}{k} \tag{34}$$

При полноценном рассмотрении делового остатка длинной D мм и шириной S мм возможность вариантов расположения фигуры F_i увеличивается в D·S раз.

Таким образом, для достижения более оптимального результата создания карт раскроя, разрешенный угол поворота фигуры должен быть меньше большего угла ГО.

2.5 Метод оптимизации «рюкзака»

В данной работе рассматривается метод составление карт раскроя ДО нестандартной формы, основанный на задаче о рюкзаке с применением жадного алгоритма.

Имеется листы ДО нестандартной формы в количестве m штук с площадью S мм^2. Необходимо разместить на данном количестве ДО n-видов ГО i-ого (i=1..n) типа с площадью S={s_i, i=1..n} мм^2 в количестве K={k_i, i=1..n} штук. W – вес делового остатка. Находится по формуле (35).

$$W=S \tag{35}$$

Каждый ГО так же имеет свой вес W_i и стоимость V_i. Находятся по формуле (36-37).

$$W_i=S_i \tag{36}$$

$$V_i= \max \{ \max C_{j1} , \max C_{j2} \} , j=1..t_i, i=1..n \tag{37}$$

Относительная стоимость единицы ГО i-ого типа [P_i] считается как отношение веса ГО на его стоимость. Находятся по формуле (38).

$$P_i=\frac{W_i}{V_i} \tag{38}$$

Суть жадного метода в задаче о рюкзаке для составления карт раскроя ДО заключается в том, что ГО укладываются на ДО в порядке убывания его относительной стоимости. Другими словами, первоначально на лист ДО укладывается самый ценный ГО в необходимом количестве, таким образом, чтоб они вошли на допустимое количество листов ДО. По аналогии, размещается следующий ГО с наибольшей относительной стоимостью без учета предыдущего и так далее до тех пор, пока не будут размещены все необходимые ГО. В случае если, в отведенное количество листов ДО ГО i-ого типа не вмещается в необходимом количестве, то вмещается максимальное количество ГО данного типа.

Матрица Ksh размером 4 x n – это матрица основных данных i, W_i, P_i и k_i ГО.

Матрица сохраняется в виде (39).

$$Ksh=\begin{pmatrix} 1 & 2 & \dots & n \\ w_1 & w_2 & \dots & w_n \\ p_1 & p_2 & \dots & p_n \\ k_1 & k_2 & \dots & k_n \end{pmatrix} \quad (39)$$

Этапыработыалгоритма:

1) Сохраняем данные ГО в матрицу Ksh;

2) Отсортировываем матрицу Ksh по убыванию стоимости единицы каждого ГО [P_i] (сортировка матрицы по убыванию данных третьей строки). Алгоритмы сортировки данных подробно описываются в работе [28]. Реализовать этап можно с помощью пузырьковой строковой сортировки;

3) M=1;

4) i=1;

5) m'=1;

6) Вмещаем ГО Ksh[1,i]-ого типа в количестве Ksh[4,i] штук на лист ДО под номером m'.

k-количество размещенных ГО Ksh[1,i]-ого типа на m'-й лист ДО

7) Если k= Ksh[4,i] то если i<n то i++, k=0 и переходим на 5 этап. Иначе переход на 9 этап.

Иначе если k<Ksh[4,i] то m'++, Ksh[4,i]=Ksh[4,i]-k, k=0, переход на 6 этап.

Если m'>M то M=m'.

8) Для раскроя ГО в необходимом количестве необходимо M листов ДО.

Если M>m то можно сказать, что имеющееся количество листов ДО не достаточно для раскроя ГО в полном объеме. Напротив, если M≤m, то имеющееся количество ДО достаточно для составления карт раскроя ДО.

Таким образом, метод составление карт раскроя ДО нестандартной формы, основанный на задаче о рюкзаке с применением жадного алгоритма размещает необходимое количество ГО на ДО с учетом его веса и стоимости.

2.6 Метод комбинаторики с применением жадного алгоритма

Суть алгоритма состоят в том, что перебирается все возможные варианты составления карт раскроя ДО нестандартной формы методом его максимального заполнения, последовательно соблюдая очередность использования ГО.

Имеется листы ДО нестандартной формы с площадью S мм2. Необходимо составить k различных карт раскроя, используя n-видов ГО i-ого (i=1..n) типа с площадью $S=\{si, i=1..n\}$ мм^2.

Матрица V размером k x n – это матрица, содержащая данные очередности использования ГО при составлении карт раскроя, где k – это количество вариантов карт раскроя, n-количество типов ГО, участвующих в раскрое.

j-ая строка $V(j,)=(v_{j1}v_{j2} \ldots v_{jn})$ составляется с помощью правил комбинаторики перестановкой элементов от 1 до n без повторений, которые отличаются только порядком расположения элементов. Элементы строк матрицы V не должны повторяться. Количество всех комбинаций ищется по формуле (40). [29]

$$k=P_n=A_n^n=n! \tag{40}$$

Таким образом, при n=3 k=3!=6 матрица V принимает следующий вид:

$$V=\begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix}$$

Элемент матрицы V говорит о том, что ГО V[j,i]-ого типа в j-м варианте карты раскроя используется i-м.

Этапы алгоритма:

1) Создаем матрицу V размером k x n;
2) j=1;
3) i=1
4) Максимально размещаем ГО V[j,i]-ого типа на ДО;
5) Если i<n то i++, переход на 4 этап. Иначе переход на 6 этап;
6) Если j<k то j++, переход на 3 этап. Иначе составление карт раскроя окончено.

Таким образом, метод комбинаторики с применением жадного алгоритма реализует раскрой ДО общим способом. Минус метода в том, что карты раскроя могут повторяться, в следствии чего необходима его дополнительная проверка.

2.7 Метод комбинаторики с применением полного перебора

Суть алгоритма состоят в том, что мы заранее знаем, сколько максимум ГО каждого типа войдет на лист ДО. И методом перестановки по правилам комбинаторики составляем способы сочетания ГО на листе ДО.

Имеется листы ДО нестандартной формы с площадью S мм2. Необходимо составить k различных карт раскроя, используя n-видов ГО i-ого (i=1..n) типа с площадью S={si, i=1..n} мм2.

Массив чисел K={1,2..n} содержит данные о максимальном количестве ГО, входящих на ДО. Элемент массива k[i] говорит о том, что ГО i-ого типа входит на ДО в количестве k[i] штук.

Матрица V размером k x n – это матрица, содержащая данные о количестве ГО при составлении карт раскроя, где k – это количество вариантов карт раскроя, n-количество типов ГО, участвующих в раскрое.j-ая строка $V(j,)=(v_{j1}v_{j2}\ \ldots v_{jn})$ составляется с помощью правил комбинаторики. Каждый элемент V[j,i] должен размещаться от 0 до k[i] раз.

Количество всех комбинаций ищется по формуле (41). [29]

$$K=\prod_{i=1}^{n}(k_i+1) \tag{41}$$

Таким образом, еслиn=3 при k={1,2} K=(1+1)·(2+1)=6матрица V принимает следующий вид:

$$V=\begin{pmatrix}0 & 0\\ 0 & 1\\ 0 & 2\\ 1 & 0\\ 1 & 1\\ 1 & 2\end{pmatrix}$$

Элемент матрицы V говорит о том, что ГО i-ого типа в j-м варианте карты раскроя раскраивается V[j,i] раз.

Но при раскрое необходимо исключать строки с нулевыми элементами, и строки, в которых i-й элемент меньше k[i], а остальные элементы нулевые.

Этапы алгоритма:

1) i=1;

2) проверить сколько ГО i-ого типа разместиться на ДО, k'- максимальное количество ГО i-ого типа, размещенное на ДО;

3) k[i]=k';

4) если i<n то i++, переход на 2 этап. Иначе переход на 5 этап;

5) Создаем матрицу V размером k x n;

6) J=1;

7) I=1;

8) Размещение ГО i-ого типа V[j,i] раз;

9) Если i<n то i++, переход на 8 этап. Иначе переход на 9 этап;

10) Если j<k то j++, переход на 7 этап. Иначе составление карт раскроя окончено.

Таким образом, метод комбинаторики с применением полного перебора реализует составление всех комбинаций ГО на карте раскроя. Минус метода в том, что проверяется все варианты размещения ГО, в связи с этим затрачивается достаточно времени.

Вывод: в главе рассмотрена математическая модель составления карт раскроя ДО нестандартной формы. Алгоритм составления карт раскроя условно делится на четыре этапа: сохранение первоначальных данных в массив, составление матрицы объекта и относительно метода составления карт раскроя размещение ГО на ДО. Самый наилучший метод составления карт раскроя является метод комбинаторики с применением полного перебора, поскольку реализует все комбинации размещения ГО на ДО.

3 ТЕХНОЛОГИЯ РАЗРАБОТКИ

3.1 Анализ технических свойств системы

Система предназначена для автоматизации процесса раскроя делового остатка нестандартной формы на геометрические объекты.

Цель создания системы: Создание автоматизированной системы, реализующей карты раскроя делового остатка нестандартной формы на необходимые геометрические объекты.

Объектом автоматизации является реализация карт раскроя ДО нестандартной формы.

Функции системы:

— Составление вариантов карт раскроя методом, основанным на правилах комбинаторики с применением полного перебора;

— Составление вариантов карт раскроя методом, основанным на задаче о рюкзаке с применением жадного алгоритма;

— Составление вариантов карт раскроя методом, основанным на правилах комбинаторики с применением жадного алгоритма;

— Составление таблицы раскроя в зависимости от используемого способа раскроя, отражающая данные: вариант раскроя, количество ГО, участвующих в раскрое, площадь остатка ДО, процент заполнения ДО;

— Графическое изображение карт раскроя в двумерном пространстве, используя цветовые эффекты выделения разных видов ГО;

— Реализация карт раскроя в KOMPAS-3D v17 с целью последующей корректировки карты непосредственно пользователем (.cdw).

Автоматизированная система создания карт раскроя ДО нестандартной формы должна обеспечивать:

— Добавление данных ДО;

— Выбор данных ГО, необходимых для раскроя, из списки стандартных, с возможностью занесения необходимых размеров ГО;

— Реализацию раскроя, тремя способами (пункт 2 подпункты 2.5-2.7);

— Возможность дальнейшей корректировки карт раскроя непосредственно пользователем;

— Возможность просмотра карт раскроя в графике;

— Просмотр итоговых данных раскроя.

Система должна поддерживать пользовательский режим. В пользовательском режиме система должна выполнять все свои функции.

В программе, для поиска основных данных о фигуре, необходимо использовать основные алгоритмы вычислительной геометрии. В качестве входных данных об рассматриваемой фигуре принимаются ее координаты точек перегиба контура, другими словами координаты вершины фигуры (x,y), в следствии чего, все вычисления производятся относительно ее координат.

В качестве входных данных считаются:

— Данные делового остатка;

— Данные ГО, участвующих в составлении карт раскроя.

На выходе система получает:

— Варианты карт раскроя, реализованные методом №1, в формате .bmp и .cdw;

— Варианты карт раскроя, реализованные методом №2, в формате .bmp и .cdw;

— Варианты карт раскроя, реализованные методом №3, в формате .bmp и .cdw;

— Таблицы раскроя в зависимости от используемого способа раскроя, отражающие данные: вариант раскроя, количество шаблонов, участвующих в раскрое, площадь остатка ДО, процент заполнения ДО.

При разработке системы необходимо предусмотреть возможность корректировки описания делового остатка и конкретного ГО, так же возможность удаления ГО из списка используемых в дальнейшем раскрое.

Взаимодействие пользователя с системой должно осуществляться с помощью графического интерфейса. Графический интерфейс должен быть простым и комфортным, с необходимыми компонентами для реализации всех функций системы.

Таким образом, функции системы необходимо реализовать для 1 пользователя.

Функции пользователя:

— Работа с ДО (выбор данных ДО, сохранение ДО для раскроя, корректировка ДО);

— Работа с ГО (выбор вида ГО, заполнение данных ГО, сохранение данных ГО, корректировка данных ГО, удаление ГО из списка необходимых для раскроя);

— Выбор метода создания карт раскроя;

— Просмотр таблицы раскроя;

— Просмотр карт раскроя;

— Корректировка карт раскроя в формате файла КОМПАС-Чертеж (.cdw).

Для реализации необходимых функций описали этапы работы системы:

— Выбор данных делового остатка, его первоначальная обработка;

— Ввод данных ГО, их первоначальная обработка;

— Создание карт раскроя методом №1;

— Создание карты раскроя методом №2;

— Создание карты раскроя методом №3.

Схема работы программы второго уровня представлена в приложении Б (см. рис. Б.1).

Данные ДО загружаются в Excel файле формата (.xlsx).

Предварительная подготовка данных ДО: в ячейку A3 сохраняется количество углов ДО, в ячейку B3 сохраняется имеющееся количество ДО, в ячейку C3 сохраняется площадь фигуры ДО, в ячейку D3 сохраняется площадь листа, описывающего ДО (ширина ДО умноженная на длину ДО), в ячейках E3 и F3 сохраняется длина и ширина ДО. В ячейках G3 до [3, A3*2+6] сохраняются координаты ДО в виде массива перебора X и Y.

Пример заполнения Excel файла представлен на рисунке 25.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Основные данные ДО**						**Координаты ДО** (минимум 3 угла)							
2	**n** (кол-во углов)	**Кол-во**	**S** (фигура)	**S** (Прям-к)	**Xmax**	**Ymax**	**(x,**	**y)**	**(x,**	**y)**	**(x,**	**y)**	**(x,**	**y)**
3	4	2	2500	2500	50	50	0	0	0	50	50	50	50	0
4														

Рисунок 25 - Пример заполнения Excel файла

Поскольку данные делового остатка первоначально сохранены в Excel файл, то необходимо реализовать его поиск и выбор на ПК в программу. Далее из выбранного файла необходимо загрузить значения в массив и построить по данным матрицу делового остаткаDO. Построение матрицы ДО необходимо реализовать по описанному ранее методу (пункт 2).

Схема ввода данных ДО представлена в приложении Б (см. рис. Б.2).

ГО в программе делятся на 5 видов: треугольник, квадрат, ромб, трапеция, произвольный. В общем виде входные данные ГО представляются в виде координат вершин фигуры и необходимое кол-во шаблонов для раскроя. В случае, когда пользователю известно точный вид ГО, программа выводит способ заполнения входных данных. Таким образом, при заполнении данных шаблона вида «треугольник» достаточно

ввести ее стороны (в мм) и необходимое количество для раскроя. Аналогично для ввода данных об ромбе достаточно ввести размер стороны ромба и размер большего угла.

При вводе данных ГО, необходимых при реализации карт раскроя ДО, необходимо реализовать выбор типа ГО: треугольник, прямоугольник, ромб, трапеция, произвольная фигура. Далее необходимо реализовать возможность ввода данных ГО выбранного типа. Необходимые данные для полноценной работы программы описываются в пункте (2.1). Далее необходимо обработать введенные данные и сохранить их в массив (пункт 2.2)

Схема ввода данных ГО представлена в приложении Б (см. рис. Б.3).

Так же необходимо реализовать возможность просмотра изображения ДО и ГО, возможность просмотра их характеристик.

Реализация карт раскроя методами №1-3 осуществляется способами, описывающимися в пункте 2.3. По завершению реализации карт раскроя выбранным методом, необходимо реализовать отображения карт раскроя в программе в форматах .bmp и .cdw, так же необходимо реализовать отображения данных раскроя в таблице.

Таким образом, система должна реализовать выполнения карт раскроя ДО нестандартной формы на необходимые геометрические объекты. Рассмотрены ее функции и задачи.

3.2 Разработка интерфейса программы

Главное окно приложения представлено на рисунке 26. Компоненты окна описаны в таблице 1.

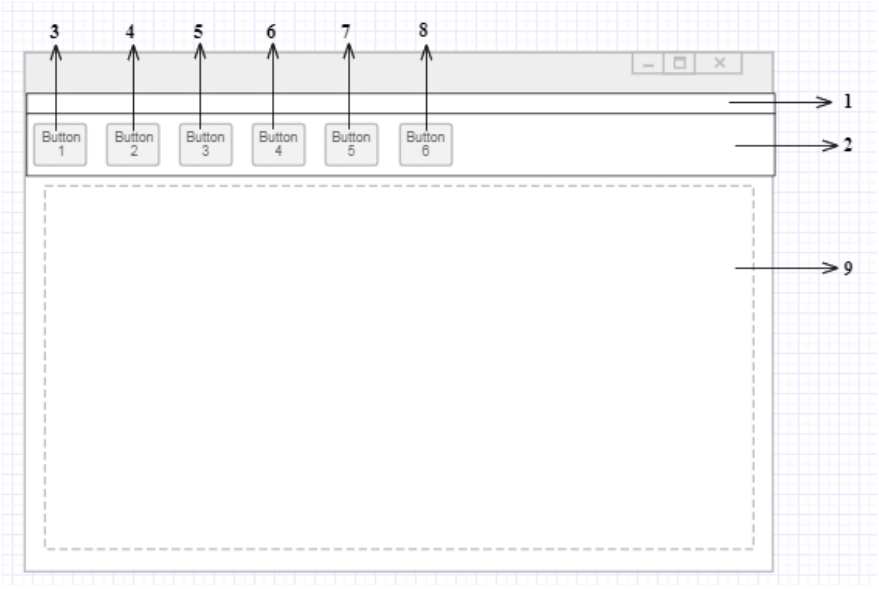


Рисунок 26 - Главное окно системы

Таблица 1 – Описание компонентов главного окна системы

№	Объект	Название	Описание
1	Menu	Навигационное меню	Представлено из 4 пунктов: проект, действие, раскрой, помощь. «Проект» - отображает основные манипуляции с проектом. Содержит подпункт «Создать новый». «Действие» - отображает основные манипуляции с входными данными. Содержит подпункты: «Добавить/изменить деловой остаток», «добавить шаблон». «Раскрой» - отображает все имеющиеся методы составления карт раскроя. Содержит подпункты: «полный раскрой», «раскрой по количеству», «общий раскрой». «Помощь» - отображает информацию по работе с программой, необходимых пользователю. Содержит подпункты: «Справка», «о программе».
2	Panel	Панель быстрого доступа	Панель быстрого доступа к инструментам программы
3	Button 1	«Новый проект»	Создает новый проект
4	Button 2	«Добавить/изменить деловой остаток»	Открывает диалоговое окно поиска ДО
5	Button 3	«Добавить ГО»	Открывает форму добавления ГО
6	Button 4	«Полный раскрой»	Раскрой методом, основанным на правилах комбинаторики с применением полного перебора

Продолжение таблицы 1.

7	Button 5	«Раскрой по количеству»	Раскрой методом, основанным на задаче о рюкзаке с применением жадного алгоритма
8	Button 6	«Общий раскрой»	Раскрой методом, основанным на правилах комбинаторики с применением жадного алгоритма
9	Frame	-	Место отображения манипуляций программы

Вид формы запуска приложения представлено на рисунке 27. Компоненты окна описаны в таблице2.

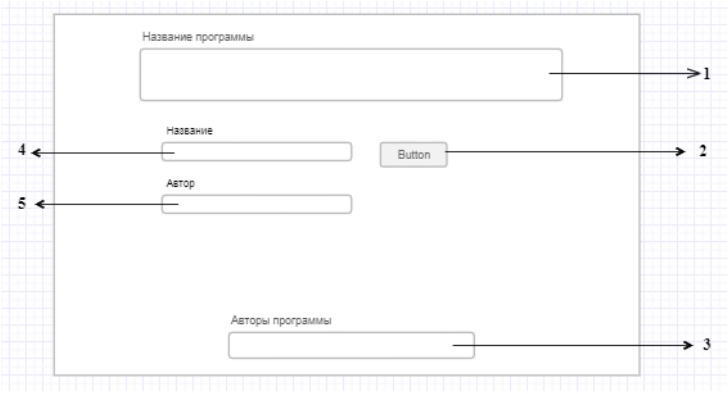


Рисунок 27 - форма ввода названия проекта

Таблица 2 - Описание компонентов окна запуска системы

№	Объект	Название	Описание
1	Static Text	Pazl-2D v1	Название системы
2	Button	ok	Кнопка подтверждения начала нового проекта
3	Static Text	Авторы: Федосеева В. А., Петренко, А. А., Зыкин С. Ю. Год создания: 2020 гг	Информация об авторах программы и года создания.
4	Edit	-	Поле ввода названия проекта
5	Edit	-	Поле ввода автора проекта

Лист отображения данных делового остатка в главном окне изображен на рисунке 28.

Компоненты листа:

1) Таблица основных данных ДО;
2) Изображение делового остатка в формате .bmp.

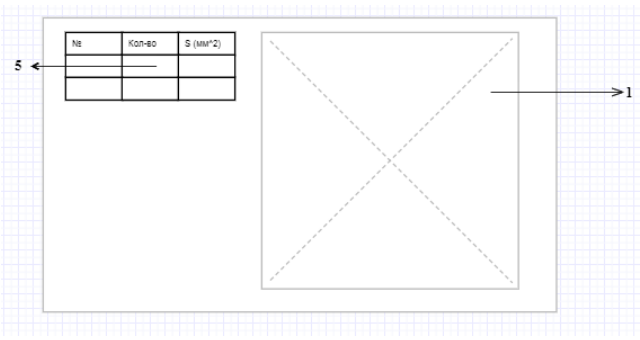


Рисунок 28 - Форма отображения данных ДО

Форма ввода данных ГО представлена на рисунке 29. Компоненты формы представлены в таблице 3.

Таблица 3 - Описание компонентов формы отображения данных ДО системы

№	Объект	Название	Описание
1	Frame	-	Форма ввода данных ГО типов: треугольник, прямоугольник, ромб, трапеция
2	Combobox	-	Выпадающий список типов ГО, содержащий пункты: треугольник, прямоугольник, ромб, трапеция, произвольный
3	Edit	-	Координаты точки перегиба контура ГО типа «произвольный»
4	Edit	-	Область ввода количества необходимых ГО
5	Button	Ok	Кнопка сохранения очередной точки перегиба контура ГО типа «произвольный»
6	Button	«Далее»	Сохранение данных ГО
7	Image	-	Изображение ГО
8	Edit	-	Область ввода размера ГО

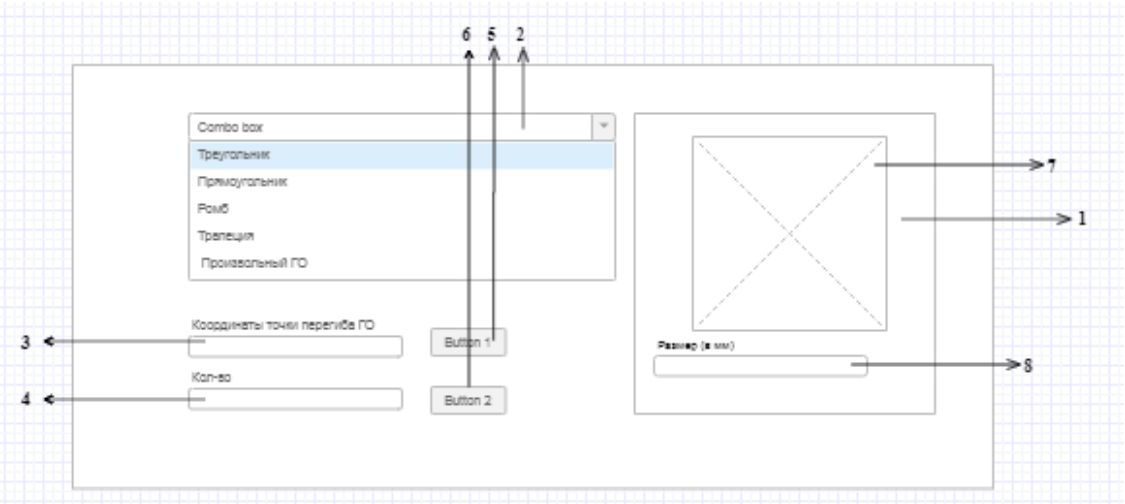


Рисунок 29 - Форма ввода данных ГО

Форма ввода данных ГО при выборе ГО типа «треугольник» представлена на рисунке 30.

Компоненты формы:

1) Область ввода размера первой стороны треугольника (мм);
2) Область ввода размера второй стороны треугольника (мм);
3) Область ввода размера третьей стороны треугольника (мм).

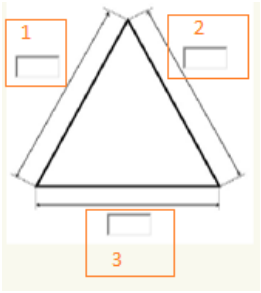


Рисунок 30 - Форма ввода данных ГО при выборе ГО типа «треугольник»

Форма ввода данных ГО при выборе шаблона типа «прямоугольник» представлена на рисунке 31.

Компоненты формы:

1) Область ввода размера первой стороны прямоугольника (мм);
2) Область ввода размера второй стороны прямоугольника (мм).

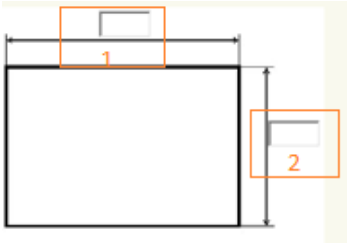


Рисунок 31 - Форма ввода данных ГО при выборе ГО типа «прямоугольник»

Форма ввода данных ГО при выборе ГО типа «ромб» представлена на рисунке 32.

Компоненты формы:

1) Область ввода размера стороны ромба (мм);
2) Область ввода размера большего, внутреннего угла ромба (град);
3) Область ввода количества необходимых ГО.

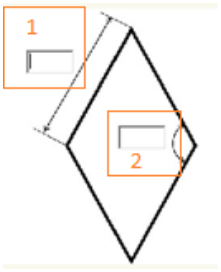


Рисунок 32 - Форма ввода данных ГОпри выборе ГОтипа «ромб»

Форма ввода данных ГО при выборе ГОтипа «трапеция» представлена на рисунке 33.

Компоненты формы:

1) Область ввода размера верхнего основания трапеции (мм);
2) Область ввода размера боковой стороны трапеции (мм);
3) Область ввода размера нижнего основания трапеции (мм);
4) Область ввода размера внутреннего угла трапеции, лежащего при основании (град).

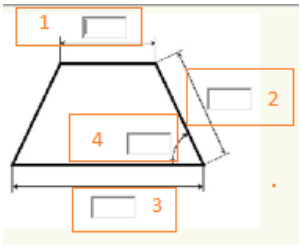


Рисунок 33 - Форма ввода данных ГО при выборе ГО типа «трапеция»

Форма отображения данных ГО в главном окне изображена на рисунке 34. Описание компонентов представлено в таблице 4.

Таблица 4 - Описание компонентов формы ввода данных ГО системы

№	Объект	Название	Описание
1	Image	-	Изображение ГО в формате .bmp
2	StringGrid	-	Таблица основных данных ГО
3	ShellListView	-	Область отображения содержимого папки C:\show projects \ < имя проекта > \ template (Шаблоны), в котором содержатся данные шаблона в формате .bmp и .cdw (КОМПАС-Чертеж)
4	Edit	Угол поворота (град)	Область ввода угла поворота фигуры при раскрое (в градусах)
5	CheckBox	Запрет на поворот прямоугольника	Область обозначения запрета/разрешения на поворот прямоугольника

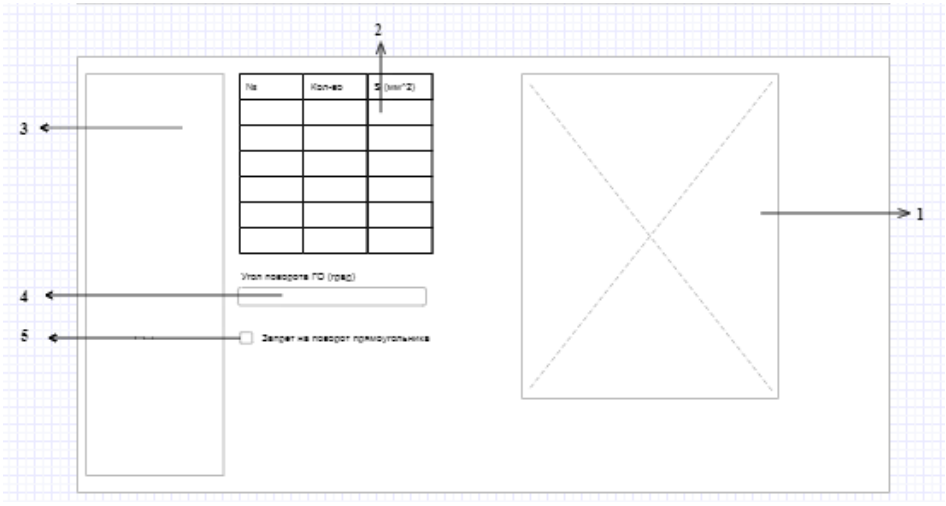


Рисунок 34 - Форма отображения данных ГО

Лист отображения результатов раскроя ДО,независимо от метода раскроя, представлен на рисунке 35.

Компоненты листа:

1) Изображение шаблона в формате .bmp;

2) Таблица основных данных шаблонов;

3) Область отображения содержимого папки C:\show projects\<имя проекта>\<название раскроя>, в котором содержатся данные шаблона в формате .bmp и .cdw (КОМПАС-Чертеж).

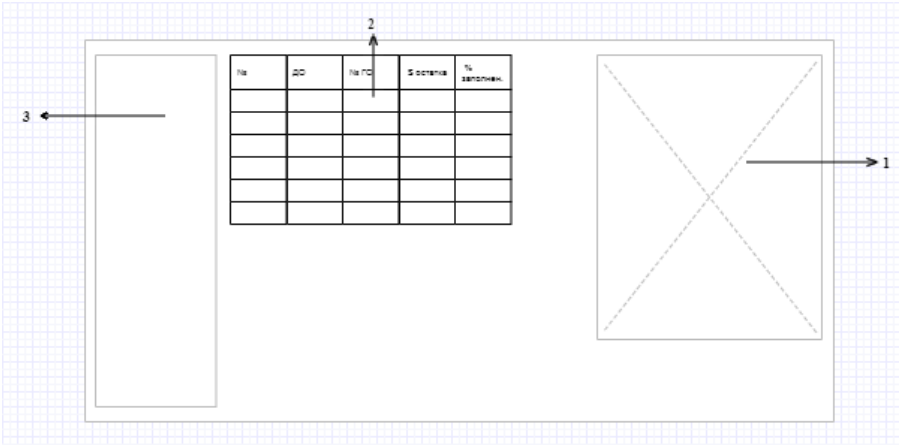


Рисунок 35 - Форма отображения данных карт раскроя

Блок схема алгоритма работы программы представлена в приложении В.

Таким образом, система должна иметь одно основное окно, в котором происходит взаимодействие пользователя с системой. Для вывода информации для пользователя необходимо восемь форм.

3.3 Сведения об ограничениях применения программы

В программе «Pazl-2Dv1» имеются следующие ограничения:

1) Единицы измерения – миллиметры;
2) Ограничения размеров по длине – от 100 мм до 500 мм;
3) Ограничения размеров по ширине – от 100 мм до 500 мм;
4) Минимальное количество вариантов ГО для раскроя – 1;
5) Максимальное количество вариантов ГО для раскроя – 5;
6) Данные ДО первоначально необходимо сохранить в Excel документе, заполненного по шаблону;
7) Координаты ДО сохраняются поочередно, по часовой стрелке, включая точку (0,0).
8) Координаты произвольного ГО сохраняются поочередно, по часовой стрелке, включая точку (0,0).

Все данные, вводимые пользователем, должны соответствовать ранее обозначенным ограничениям.

Время выполнения раскроя зависит от размера площади ДО, количества шаблонов, участвующих в раскрое и правильности введенных данных непосредственно пользователем.

При малых размерах ДО, раскрой совершается с небольшой погрешностью, обусловленной округлением данных.

Таким образом, обозначены основные ограничения и условия работы системы.

3.4 Реализация системы в визуальной среде программирования Delphi

3.4.1 Реализация компонентов интерфейса системы

Создали новый проект в Delphi 7. На Form 1 из вкладки Standard разместили компонент: MainMenu. Ввели пункта меню: проект, действие, раскрой, помощь. В пункте «проект» ввели подпункт «Новый проект», в пункте «действие» ввели подпункт «Загрузить/Изменить деловой остаток», «Добавить ГО», в пункте «раскрой» ввели подпункты «Метод №1», реализующий «Полный раскрой», «Метод №2», реализующий «Раскрой по количеству», «Метод №3», реализующий «Общий раскрой». В пункте «помощь» ввели подпункты «Руководство пользователю» и «о программе». Свойства компонента MainMenu представлены в таблице5.

Таблица 5 - Свойства компонента MainMenu

Выделенный объект	Вкладка окна Object Inspector	Caption	Name	Hint	ShowHint	Butmap
Form1.MainMinu1 (Вкладка Standard)	Properties	Проект	A1	-	False	-
		Новый проект	N2	-	False	
		Действие	N4	-	False	
		Загрузить/Изменить деловой остаток	N5	-	False	
		Добавить ГО	N6	-	False	
		Раскрой	N9	-	False	
		Метод №1	N10	Раскрой методом, основанным на правилах комбинаторики с применением полного перебора	True	
		Метод №2	N11	Раскрой методом, основанным на задаче о рюкзаке с применением жадного алгоритма	True	
		Метод №3	N12	Раскрой методом, основанным на правилах комбинаторики с применением жадного алгоритма	True	
		Помощь	N1	-	False	
		Руководство пользователю	N7	-	False	
		О программе	N8	-	False	

Из вкладки Standard разместили компонент: Panel. На компоненте Panel из вкладки Additional разместили 6 компонентов SpeedButton. Свойства компонентов представлены в таблице 6.

Таблица 6 - Свойства компонентов на Panel

Выделенный объект	Вкладка окна Object Inspector	Name	Hint	ShowHint	Glyph	Color
Form1.Panel.SpeedButton1	Properties	SpeedButton1	Загрузить деловой остаток	True		$00EEF9F9
Form1.Panel.SpeedButton2	Properties	SpeedButton2	Добавить ГО	True		$00EEF9F9
Form1.Panel.SpeedButton3	Properties	SpeedButton3	Раскрой методом, основанным на правилах комбинаторики с применением полного перебора	True		$00EEF9F9
Form1.Panel.SpeedButton4	Properties	SpeedButton4	Раскрой методом, основанным на задаче о рюкзаке с применением жадного алгоритма	True		$00EEF9F9
Form1.Panel.SpeedButton5	Properties	SpeedButton5	Раскрой методом, основанным на правилах комбинаторики с применением жадного алгоритма	True		$00EEF9F9
Form1.Panel.SpeedButton7	Properties	SpeedButton7	Новый проект	True		$00EEF9F9

Для регистрации нового проекта создали Frame1 (File/new/Frame). Расположили Frame1 на Form 1.

На Frame1 из вкладки Standard разместили 5 компонентов StaticText, 2 компонента Edit. Из вкладки Additional разместили компонент BitBtn. Свойства компонентов представлены в таблице 7.

Таблица 7 – Свойства компонентов формы регистрации нового проекта

object	Left	Top	Width	Height	Align	Alignment	Caption
object StaticText29: TStaticText	0	405	643	17	alBottom	taCenter	Год создания: 2020 г.
object StaticText27: TStaticText	0	388	643	17	alBottom	taCenter	Авторы: Федосеева В. А., Зыкин С. А., Петренко А. А.
object StaticText26: TStaticText	0	0	643	61	alTop	taCenter	'"Pazl-2D v1"'
object StaticText2: TStaticText	201	200	49	23	alNone	taLeftJustify	Автор
object StaticText1: TStaticText	178	177	72	23	alNone	taLeftJustify	Название
object Edit2: TEdit	267	200	121	21	-	-	-
object Edit1: TEdit	267	176	121	0	-	-	-
object BitBtn6: TBitBtn	395	173	81	25	-	-	Ок

Вид окна при запуске приложения представлено на рисунке 36.

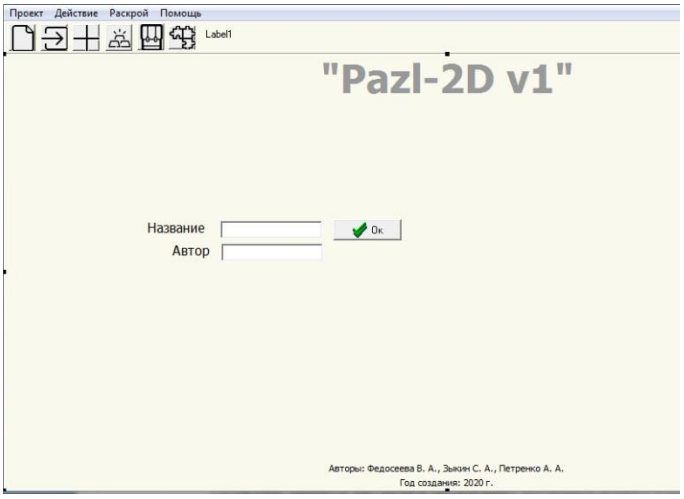


Рисунок 36 - Вид окна при запуске приложения

Листинг Frame1 в приложении Г.

Создали Frame2 для отображения входных и выходных данных приложения. Так же расположили Frame2 на Form 1. На Frame2 из вкладки Win32 разместили компонент PageControl. Свойства расположение компонента Aligh: alClient. В PageControl создали 5 страниц отображения данных: сырье, ГО, раскрой методом №1, раскрой методом №2, раскрой методом №3.

На лист отображения данных сырья (ДО) из вкладки Additional добавили 2 компонента Image, 1 компонент StringGrid. Изображение листа «Сырье» представлено на рисунке 37.

Рисунок 37 - Изображение листа «Сырье»

На лист отображения геометрических объектов из вкладки Additional добавили 2 компонента Image, 1 компонент StringGrid, так же расположили компоненты: StaticText, ShellListView, CheckBox, LabeledEdit, Splitter. Изображениелиста «ГО» представленонарисунке38.

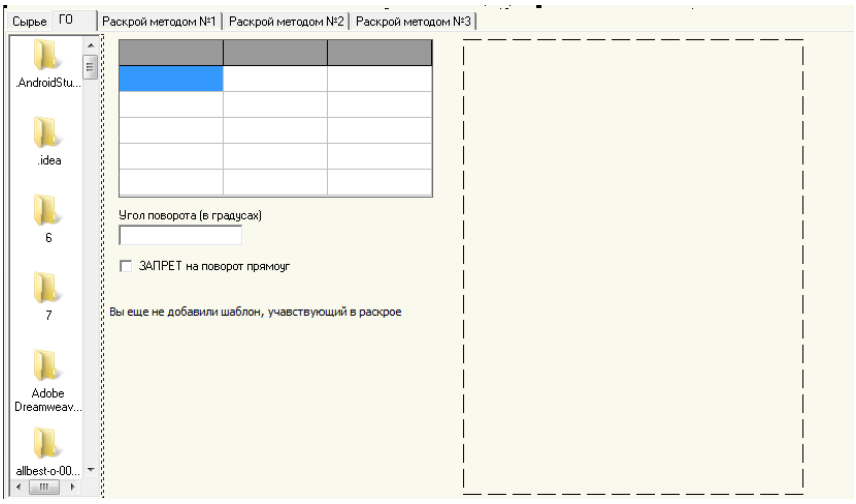


Рисунок 38 - Изображение листа «ГО»

Листы отображения данных раскроя созданы по аналогии. Налистахразмещенькомпоненты: Image, StringGrid, StaticText, ShellListView, Splitter. Изображение листа «Раскрой методом №1» представлено на рисунке39.

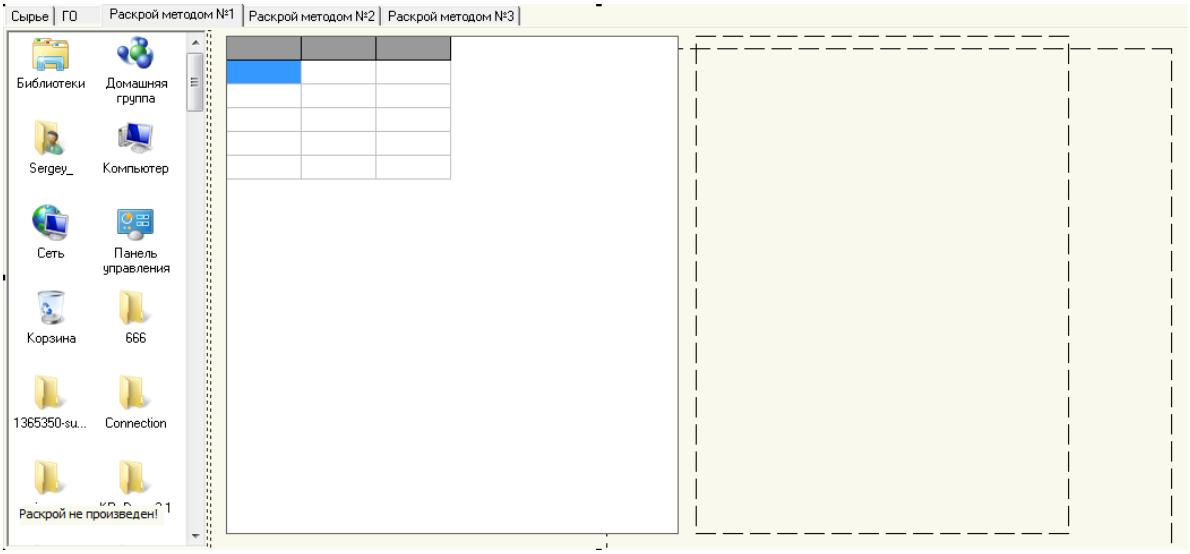


Рисунок 39 - Изображение листа «Раскрой методом №1»

Листинг Frame 2 отображения входных и выходных данных системыпредставлен в приложении Д.

Создали Frame4 для добавления данных геометрических объектов, используемых в раскрое. Так же расположили Frame4 на Form 1.

На Frame4 для выбора типа ГО из вкладки Standard расположили 2 компонента Button и компонент ComboBox с свойством items: треугольник; прямоугольник; ромб; трапеция; произвольный. Так же из вкладки Additional расположили 3 компонента LabeledEdit и BitBtn, необходимых для ввода координат произвольного ГО и необходимого количества.

Для ввода данных таких ГО как треугольник, прямоугольник, ромб и трапеция создали отдельные Frame с вводом необходимых параметров.

Создали Frame для ввода данных треугольника. Расположили компонент Image и 3 компонента Edit для ввода сторон треугольника. Создали Frame для ввода данных прямоугольника. Расположили компонент Image и 2 компонента Edit для ввода сторон прямоугольника. Создали Frame для ввода данных ромба. Расположили компонент Image и 2 компонента Edit для ввода стороны и угла ромба. Создали Frame для ввода данных трапеции. Расположили компонент Image и 4 компонента Edit для ввода трех сторон трапеции и угла нижнего основания. Изображение Frame 6-9 представлено на рисунке40.

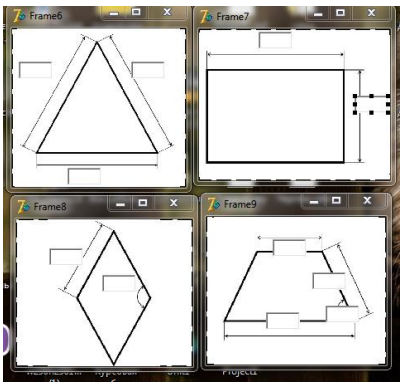


Рисунок 40 - Изображение Frame 6-9

Изображение Frame4 представлено на рисунке41.

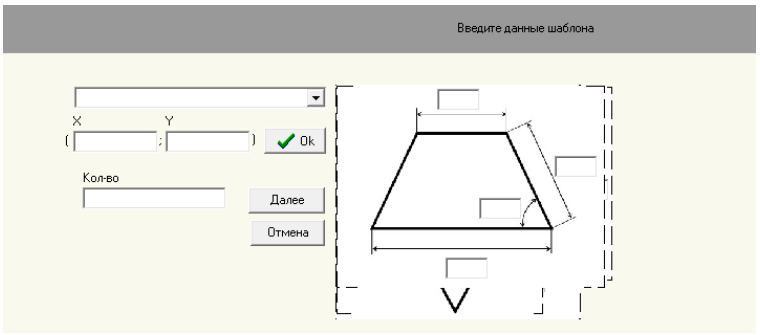


Рисунок 41 - Форма ввода данных ГО

Листинг Frame 4для добавления данных геометрических объектов, используемых в раскрое в приложении Е.

Создали Frame для визуализации этапов составления карт раскроя. Расположили Frame на Form 1. Расположили 5 компонентов Label, компонент ProgressBar, и 2 компонента StaticText. Код свойств компонентов представлено в приложение Ж.

Frameпредставленовнарисунке42.

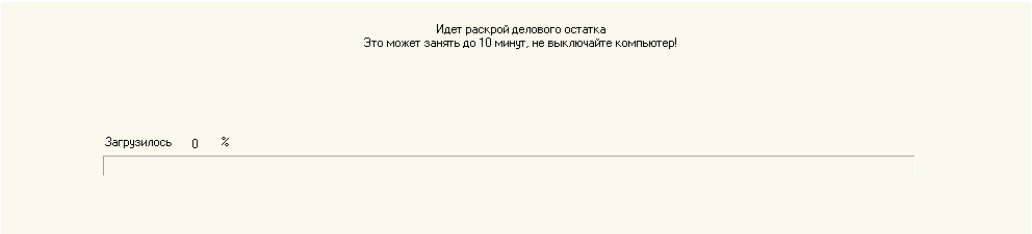


Рисунок 42 - Форма отображения статуса создания карт раскроя

Создали Form2 с данными «о программе». Расположили 8 компонентов StaticText, компонент Image и Panel. Изображение формы на рисунке 43.

Рисунок 43 - Форма отображения данных "О программе"

3.4.2 Описание реализации основных функций системы

При реализации действий сохранили основные атрибуты системы, необходимые для реализации методов:

- Data: array of Real;
- Matr: array of array of integer;
- DataF: array [1..1000,1..1000] of real;
- F: array [1..1000,0..1600] of real;
- MatrSH : array [-1000..1000,-1000..1000] of integer;
- Raskroi: array [1..1500,0..15] of double;
- STabl0: array [0..100,0..1000] of double;
- STabl2: array of array of double;
- Raskroi2: array of array of double;
- STabl3: array of array of double;
- Raskroi3: array of array of double;
- STabl4: array of array of double;
- Raskroi4: array of array of double.

Где Data – одномерный массив, содержащий основные данные делового остатка. Элементы массива Data представлены в таблице 8.

Таблица 8 – Элементы массива Data

Наименование	Содержание
Data [0]	кол-во углов ДО
Data [1]	Кол-во ДО данного вида
Data [2]	Площадь ДО ($мм^2$)
Data [3]	Площадь исходного листа
Data [4]	Длина листа
Data [5]	Ширина листа
Data [6]	Координаты 1 точки ДО по оси X
Data [7]	Координаты 1 точки ДО по оси Y
…	
Data [Data[0]*2+4]	Координаты n-ой точки ДО по оси X
Data [Data[0]*2+5]	Координаты n-ой точки ДО по оси Y

$Matr_{SxD}$ – матрица делового остатка размером SxD, где S=Data [5], D=Data [4]

DataF – матрица основных данных ГО размером nx8, где n – количество ГО, участвующих в раскрое. Элементыматрицы DataF представлены в таблице 9.

Таблица 9 – Элементы матрицы DataF

№ ГО	I, i=1..n
Data [i][1]	Площадь i-ого ГО
Data [i] [2]	Минимальная координата i-ого ГО по оси X
Data [i] [3]	Максимальная координата i-ого ГО по оси X
Data [i][4]	Минимальная координата i-ого ГО по оси Y
Data [i][5]	Максимальная координата i-ого ГО по оси Y
Data [i][6]	Площадь исходного листа i-ого ГО
Data [i][7]	Необходимое количество i-ого ГО
Data [i][8]	Номер типа ГО

F – массив координат ГОразмером nx100, где n – количество ГО, участвующих в раскрое. Элементы матрицы F представлены в таблице 10.

Таблица 10 – Элементы матрицы F

№ ГО	I, i=1..n
F [i][0]	t-Количество точек перегиба контура i-ого ГО
F [i][1]	Координата 1-ой точки перегиба ГО по оси X
F [i][2]	Координата 1-ой точки перегиба ГО по оси Y
…	
F [i][t*2-1]	Координата t-ой точки перегиба ГО по оси X
F [i][t*2]	Координата t-ой точки перегиба ГО по оси Y

Массивы STabl и Raskroi необходимы для реализации карт раскроя.

При открытии приложения размер Form 1 определяется по правилу золотого сечения по формуле (42).

$$\frac{A}{B}=0{,}666 \quad (42)$$

где A – длина формы;

B – ширина формы.

Если B=681, то $A = B \cdot 0.666 = 450$

При открытии приложения доступна Frame21, остальные компоненты недоступны. Кнопка сохранения недоступна до тех пор, пока не введено корректное название, ввод автора недоступно до тех пор, пока не введено название. Фрагмент кода, описывающее открытие приложения представлено в листинге 1.

Листинг 1:

```
Frame21.Visible:=true;
Frame21.Edit2.Enabled:=false;
Frame21.StaticText2.Enabled:=false;
Frame21.BitBtn6.Visible:=true;
Frame21.BitBtn6.Enabled:=false;
Frame101.Visible:=false;
  Frame31.Visible:=false;
  Panel1.Enabled:=true;
  SpeedButton5.Enabled:=false;
  frame41.Visible:=false;
  SpeedButton1.Enabled:=false;
  SpeedButton2.Enabled:=false;
  SpeedButton3.Enabled:=false;
  SpeedButton4.Enabled:=false;
```

Фрагмент кода, реализующего проверку на ввод названия, представлен в листинге2.

Листинг2:

```
procedure TForm1.Frame21Edit1Change(Sender: TObject);
begin
  if Frame21.edit1.text="
  then
  begin
    Frame21.BitBtn6.Enabled:=false;
  Frame21.Edit2.Enabled:=false;
    Frame21.StaticText2.Enabled:=false;
  end
  else
  begin
    Frame21.Edit2.Enabled:=true;
    Frame21.StaticText2.Enabled:=true;
Frame21.BitBtn6.Enabled:=true;
```

```
end;
end;
```

Фрагмент кода, реализующего перевод курсора с названия на автора, представлен в листинге 3.

Листинг 3:

```
procedure TForm1.Frame21Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if not (key in [#13, #8,'a'..'я','.','A'..'Я','0'..'9',' ']) then key:=#0;
if key=#13 then Frame21.Edit2.SetFocus;  //Если нажата клавиша Enter//To переместить курсор во второе поле
end;
```

Вид окна при запуске приложения представлено на рисунке 44.

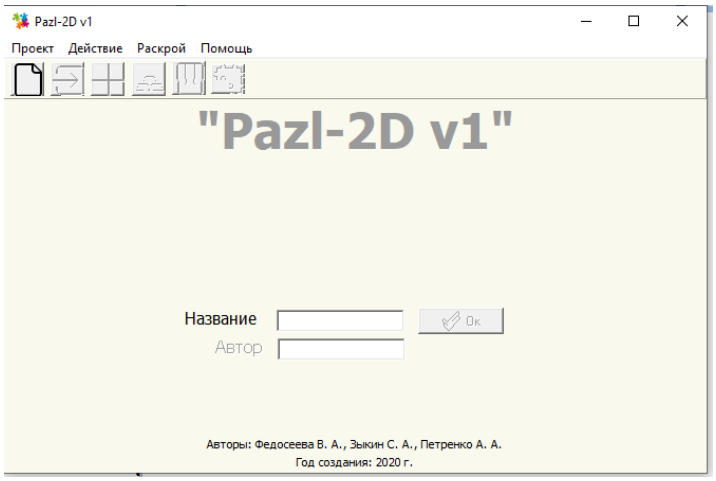


Рисунок 44 - Вид окна запуска приложения

При нажатии кнопки «ОК» создается каталог на диске С для хранения всех данных об проекте с именем, сопоставляемым с названием проекта. Далее закрывается Frame21, панель кнопок разблокируется. Разрешается добавлять данные делового остатка, все остальные действия заблокированы. Код кнопки «ок» представлен в листинге 4.

Листинг4:

```
procedureTForm1.Frame21BitBtn6Click(Sender: TObject);
begin
  ProgDIR:='C:';//ExtractFilePath('Project1.exe');
  if DirectoryExists(ProgDIR+'\show projects'+'\'+Frame21.edit1.text)
then ShowMessage('Проект с таким именем уже существует')
else
  begin
    ForceDirectories(ProgDIR+'\show projects'+'\'+Frame21.edit1.text);
    q:=true;
    Frame21.Visible:=false;
    Panel1.Enabled:=true;
```

```
SpeedButton1.Enabled:=true;
  SpeedButton2.Enabled:=false;
SpeedButton3.Enabled:=false;
SpeedButton4.Enabled:=false;
SpeedButton5.Enabled:=false;
Frame31.Visible:=false;
   Frame31.PageControl1.ActivePageIndex:=0;
end;
end;
```

Главное рабочее окно программы представлено на рисунке 45.

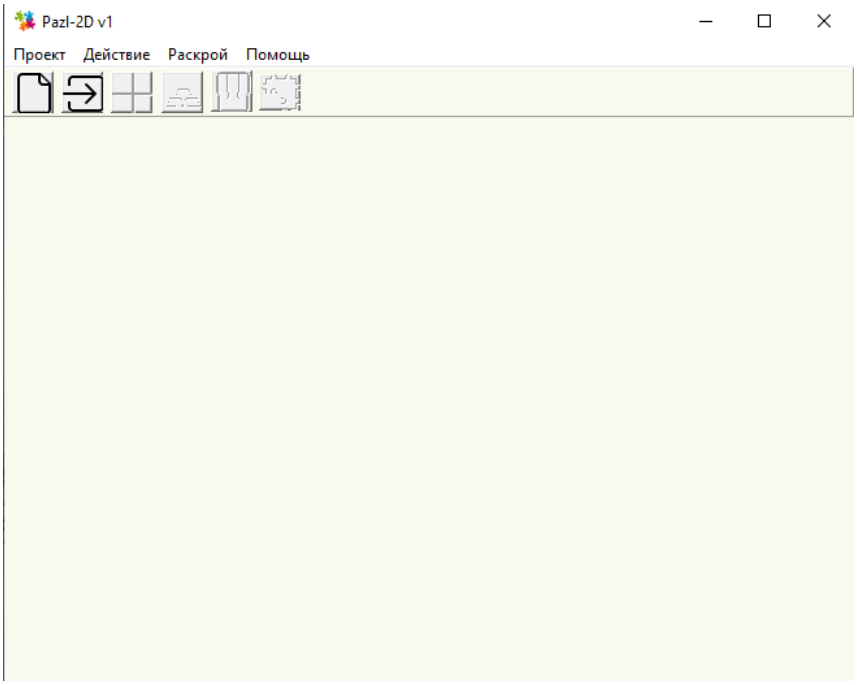


Рисунок 45 - Главное рабочее окно программы

При нажатии на иконку , расположенную в верхней панели инструментов,либо нажать «действие»/«загрузить/изменить деловой остаток» открывается диалоговое окно, где необьходимо выбрать файлExcel с данными ДО. Фрагмент кода, отвечающего за открытие диалогового окна ДО представлен в листинге 5.

Листинг5:

```
ifOpenDialog1.Execute
then
begin
if not FileExists(OpenDialog1.FileName)
then
   begin
     MessageBox(Handle, 'Файл с заданным именем не найден! Действие отменено.',
     'Внимание!', MB_OK + MB_ICONWARNING + MB_APPLMODAL);
Exit;
   end
```

```
    else
    begin
FilePath:=OpenDialog1.FileName;//сохранение пути к файлу
End;
```

В случае, если файл успешно найден создание OLE объект, данные извлекаются в массив Data[i], закрывается OLE объект и файл. Фрагмент кода, отвечающего за загрузку данных ДО представлен в листинге 6.

Листинг 6:

```
//открытие выбранного файла
    FilePath:=OpenDialog1.FileName;
//1.Открытие Excel файла
    E:=CreateOleObject('Excel.Application');
E.Workbooks.Open(FilePath);//открытие  (содержится в корневом каталоге)
    //2.1 Сохранение данных ДО  из EXCEL в Data[i]
    n:=E.ActiveWorkBook.WorkSheets.Item[1].Cells[3,1]; //сколько всего точек в ДО
n:=n*2+6;
    SetLength(Data,n);
For i:=0 to (n-1) do data[i]:=E.ActiveWorkBook.WorkSheets.Item[1].Cells[3,i+1];
//3.Закрытие Excel файла
    E.quit;
```

Диалоговое окно поиска файла с данными делового остатка на рисунке46.

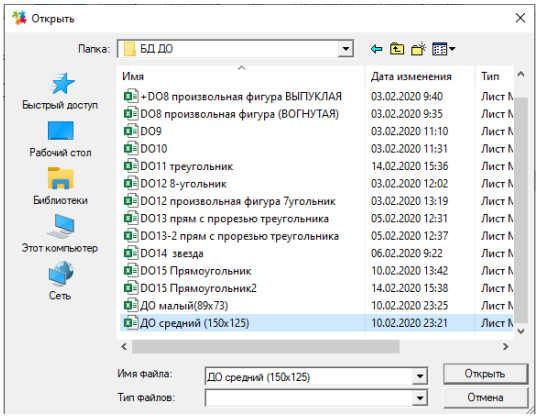


Рисунок 46 - Диалоговое окно поиска файла с данными делового остатка

Далее формируется матрица делового остатка Matr[i][j].

Для этого заранее создали переменные m,j,x,y,g,g1,y2,x2,g2,x1 типа Integer, где n и m-размер матрицы. F11 типа Boolean. Xmin, Xmax, Ymin, Ymax, j45 типа real. Так же массив с координатами фигуры Z: array [0..100,0..100] of real. Фрагмент кода выделение области памяти для матрицы представляется в листинге 7.

Листинг 7:

```
n:=Trunc(Data[5]); //Ymax+1 ткеще 0
m:=Trunc(Data[4]); //Xmax+1 ткеще 0
```

```
SetLength(Matr,n+1,m+1);
```

Фрагмент кода сохранения матрицы ДО с нулевыми данными представлено в листинге 8.

Листинг8:

```
For i:=0 to (n) do
      For j:=0 to (m) do
Matr[i][j]:=0;
```

Фрагмент кода сохранения данных в дополнительный массив для работы представлен в листинге 9.

Листинг9:

```
j:=1;
      for g:=6 to (n-1) do
      begin
        Z[0,j]:=data[g];
        j:=j+1;
end;
```

Фрагмент кода поиска уравнений прямых в фигуре ДО представлен в ПРИЛОЖЕНИИ З.

Фрагмент кода поиска матрицы фигуры ДО по найденным координатам представлен в приложение И.

При полноценном сохранении всех необходимых данных изображение ДО рисуется в КОМПАС и в image, так же сохраняются данные в таблице StringGrid3.

Фрагмент кода создания объекта автоматизации KOMPAS_Graphic представлен в приложение К (п. К.1).Фрагмент кода рисования ДО представлен приложение К (п. К.2). Далее добавление ДО блокируется, и разблокировается возможность добавления данных ГО. Frame31 становится доступным для пользователя. Фрагмент кода закрытия компас представлен в приложение К (п. К.3).

Листинг кнопки добавления ДО в приложении П.

Отображение открывшегося окна изображено на рисунке 47.

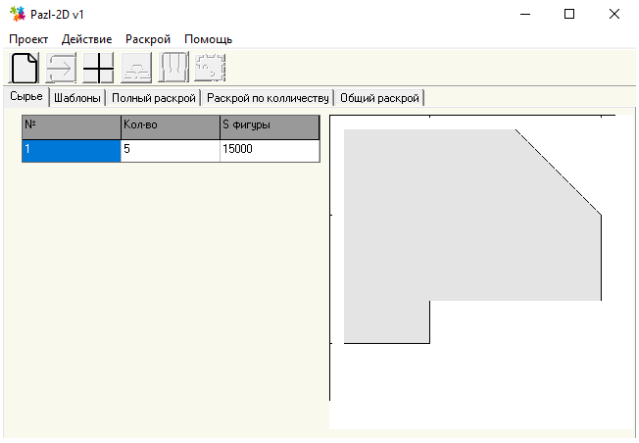


Рисунок 47 - Лист отображения делового остатка в главном окне

При нажатии на иконку ┼, расположенную в верхней панели инструментов, либо нажать «действие/Добавить шаблон»Frame31 становится не видимым, frame41 доступно для пользователя. Так же, поскольку разрешается максимум 5 ГО в раскрое, то происходит накапливание количества ГО в переменную n целочисленного типа. Листинг процедуры добавления ГО представлен в приложении Р. Начальная форма ввода данных шаблонов на рисунке 48.

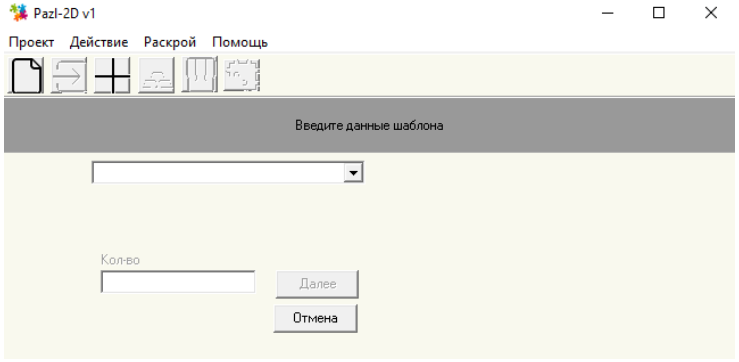


Рисунок 48– Начальная форма ввода данных шаблонов

Первоначально подразумевается то, что пользователь выбирает тип ГО в ComboBox, необходимого для составления карт раскроя. Вид выпадающего списка типов ГО представлено на рисунке 49.

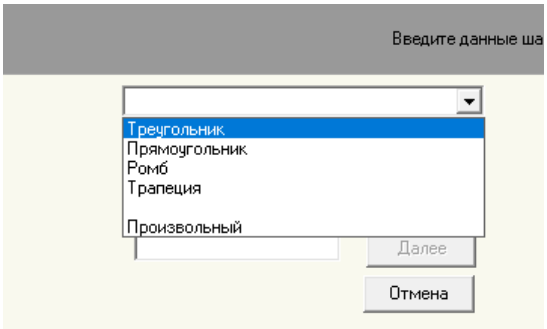


Рисунок 49 - Вид выпадающего списка типов ГО

При выборе определенного типа становится доступны различные компоненты frame41, отвечающие за данный тип ГО. При выборе типа ГО «треугольник» доступен Frame61, при выборе типа ГО «прямоугольник» доступен Frame71, при выборе типа ГО «ромб» доступен Frame81, при выборе типа ГО «трапеция» доступен Frame91. При выборе типа ГО «произвольный» доступны компоненты ввода координат произвольной фигуры. Все стороны ГО указываются в мм, углы указываются в градусах. Фрагмент кода сохранения координат и всех необходимых входных данныхотносительно его типа представлен в приложении Л.Сохранение координат ГО и первоначальных данных в массивы происходит по алгоритму, описанному в пункте (2.2). Первоначально происходит проверка корректности данных, затем сохранение данных в массив.

При полноценном сохранении всех необходимых данных изображение ГО рисуется в КОМПАС и в image, изображения сохраняются в каталог C:\\show projects\<Имя проекта>\template(Шаблоны)\'+IntToStr(n)+'<формат>', так же сохраняются данные в таблице StringGrid. Фрагмент кода закрытия frame41 представлен в листинге 10.

Листинг 10:

```
frame31.Enabled:=true;// действия в листах
Panel1.Enabled:=true;//запрет на действия в листах
frame41.Visible:=false;//видимость панели ввода шаблонов
```

Пример отображения сохраненного ГО треугольник в главном окне на рисунке 50.

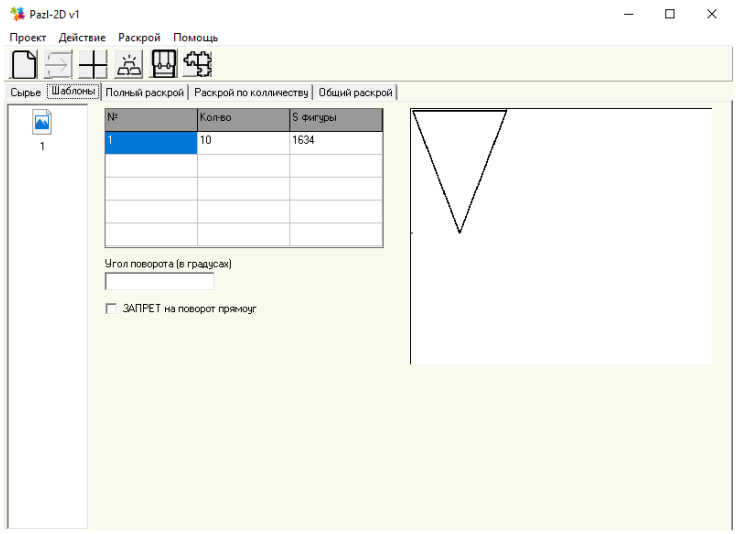


Рисунок 50 - Лист отображения сохраненного ранее шаблона в главном окне

Реализовали создание карт раскроя методом, основанным на задаче о рюкзаке с применением жадного алгоритма согласно разработанному ранее алгоритму (пункт 2.5). Отрывок кода, отвечающего за сохранение матрицы Ksh и ее сортировка представлено в листинге 11.

Листинг11:

```
for j:=1 to 5 do
  begin
    Ksh[1,j]:=0;
    Ksh[3,j]:=0;//Wi = S ВЕС
    Ksh[2,j]:=0;
Ksh[4,j]:=0;
end;
for j:=1 to Shabl do
  begin
    Ksh[1,j]:=j;
    Ksh[3,j]:=Trunc(DataF[j,1]);//Wi = S ВЕС
    if DataF[j][3]>DataF[j][5] then Ksh[2,j]:=Trunc((DataF[j,1]/DataF[j][3]))//Ui = max/S
СТОИМОСТЬ
else Ksh[2,j]:=Trunc((DataF[j,1]/DataF[j][5]));
    Ksh[4,j]:=Trunc(DataF[j][7]); //кол-во
    m:=m+Trunc(DataF[j][7]);  { }
end;
  {сортировка данных Пузырьковая сортировка в Delphi.}
if Shabl>1 then
  begin
  for i:=1 to Shabl do
    for j:=1 to Shabl-i do
      if Ksh[2,j]<Ksh[2,j+1] then
      begin {Обменэлементов}
        j3:=Ksh[2,j];
        Ksh[2,j]:=Ksh[2,j+1];
        Ksh[2,j+1]:=j3;
        j3:=Ksh[1,j];
        Ksh[1,j]:=Ksh[1,j+1];
        Ksh[1,j+1]:=j3;
        j3:=Ksh[3,j];
        Ksh[3,j]:=Ksh[3,j+1];
        Ksh[3,j+1]:=j3;
        j3:=Ksh[4,j];
        Ksh[4,j]:=Ksh[4,j+1];
        Ksh[4,j+1]:=j3;
      end;
  end;
```

Отрывок кода, реализующий алгоритм создания карт раскроя данным методом представлен в приложении М.

Иллюстрация работы алгоритма при ГО типа «треугольник» с размерами 50х70х70 мм в количестве 10 штук и ГО типа «прямоугольник» размером 40х80 мм в количестве 10 штук и при повороте на 15 градусов представлено на рисунках 51-52.

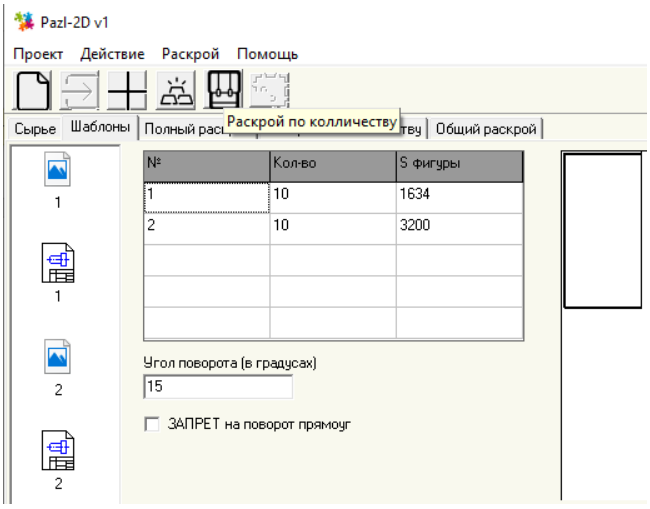


Рисунок 51 - Лист отображения ГО

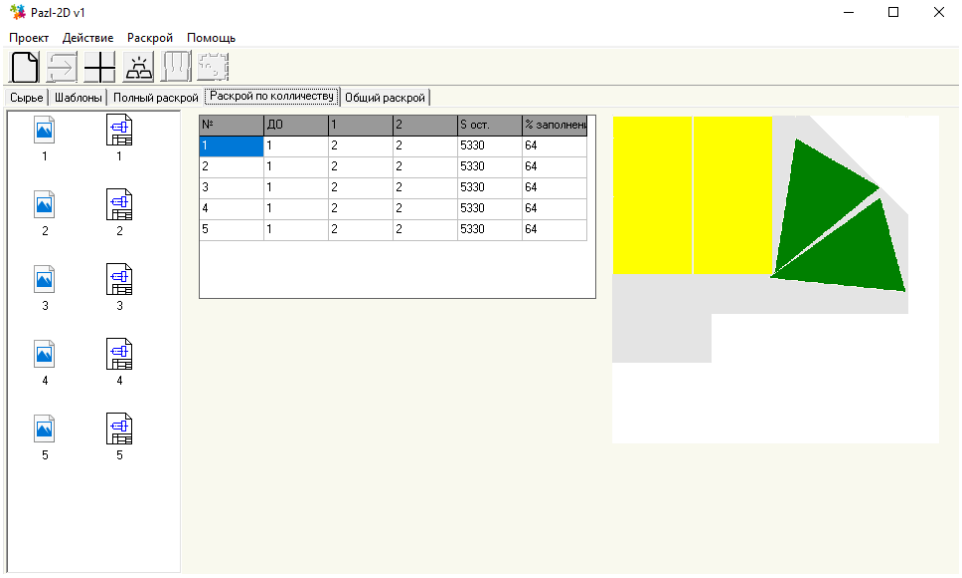


Рисунок 52 - Лист отображения данных раскроя

После реализации метода отображается диалоговое окно, в котором выводится количество листов, необходимых для раскроя ГО (см. рисунок 53).

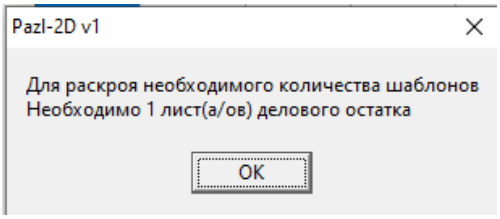


Рисунок 53 - Диалоговое окно

В папке C:\showprojects\Проект 01\ Nesting №3 сохраняются все карты раскроя для дальнейшего манипулирования (см. рисунок 54).

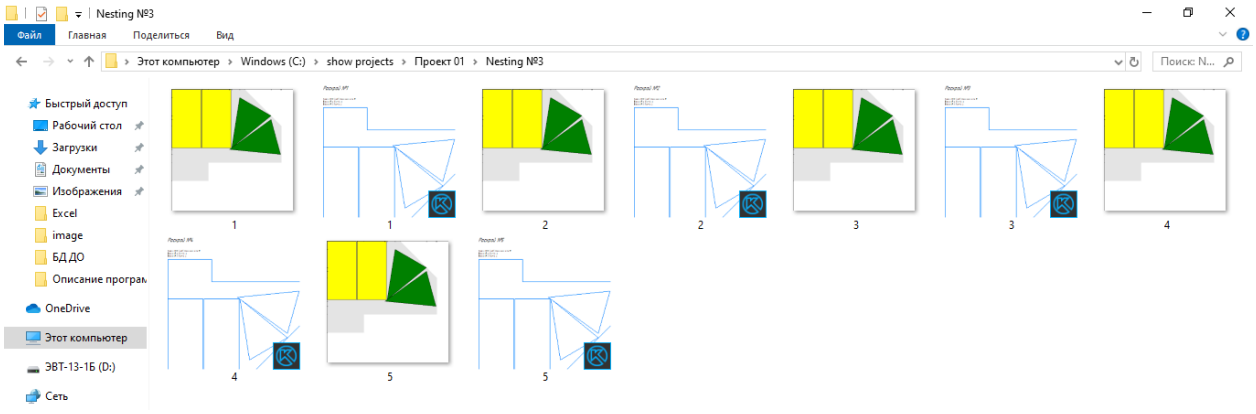


Рисунок 54 - содержимое папки C:\show projects\Проект 01\ Nesting №3

Реализовали создание карт раскроя методом, основанным на правилах комбинаторики с применением жадного алгоритма согласно разработанному ранее алгоритму (п. 2.7).

Отрывок кода, реализующий алгоритм создания карт раскроя данным методом представлен в приложении Н.

Иллюстрация работы алгоритма при ГО типа «треугольник» с размерами 50x70x70 мм и ГО типа «прямоугольник» размером 40x80 мм и при повороте на 15 градусов представлено на рисунках 55.

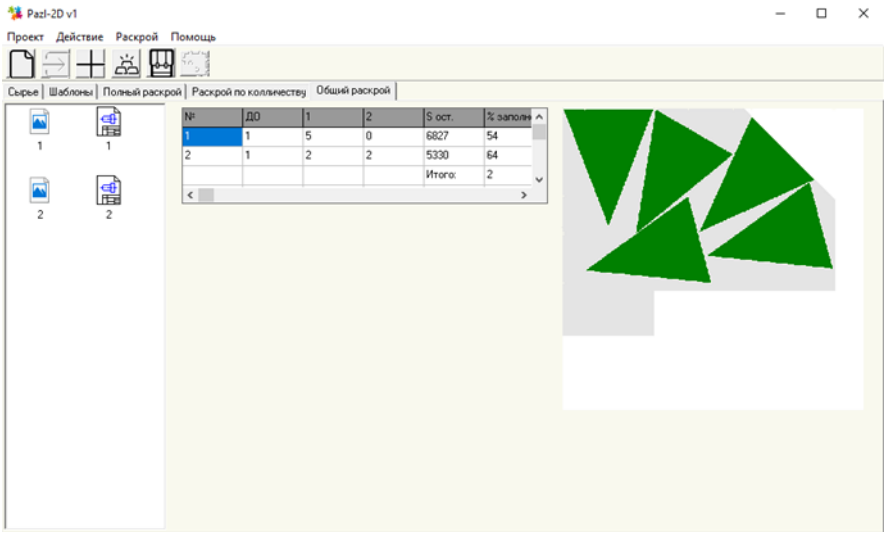


Рисунок 55 - Лист отображения данных раскроя

В папке C:\showprojects\Проект 01\ Nesting №4 сохраняются все карты раскроя для дальнейшего манипулирования (см. рисунок 56).

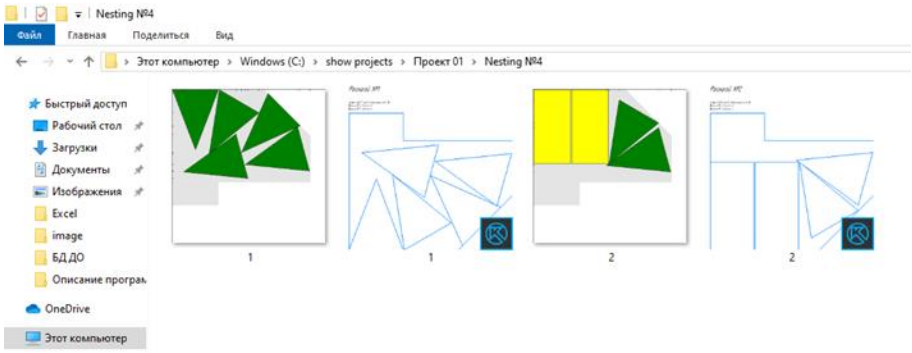


Рисунок 56 - содержимое папки C:\show projects\Проект 01\ Nesting №3

Реализовали создание карт раскроя методом, основанным на правилах комбинаторики с применением полного перебора согласно разработанному ранее алгоритму (пункт 2.8).

Отрывок кода, реализующий алгоритм создания карт раскроя данным методом представлен в приложении О.

Иллюстрация работы алгоритма при ГО типа «треугольник» с размерами 50x70x70 мм и ГО типа «прямоугольник» размером 40x80 мм и при повороте на 15 градусов представлено на рисунках 57.

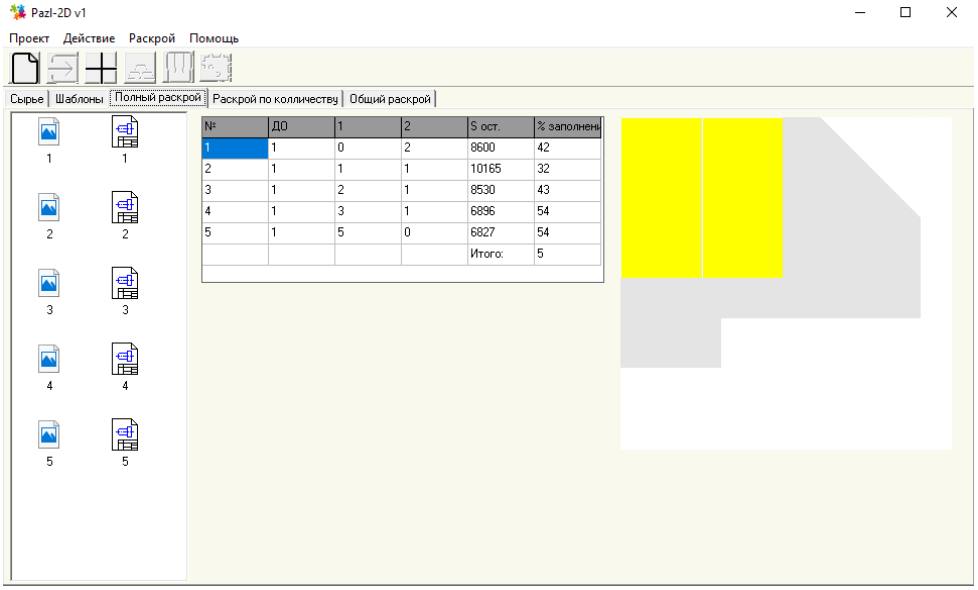


Рисунок 57 - Лист отображения данных раскроя

В папке C:\showprojects\Проект 01\ Nesting №2 сохраняются все карты раскроя для дальнейшего манипулирования (см. рисунок 58).

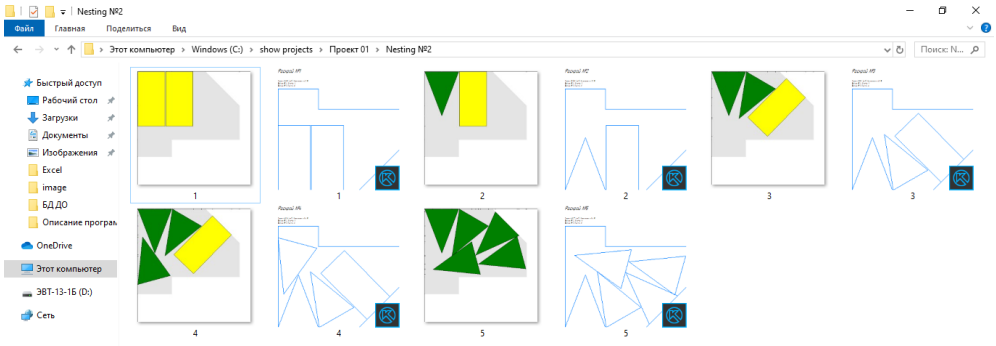


Рисунок 58 - Содержимое папки C:\show projects\Проект 01\ Nesting №2

Edit и LabeledEdit защитили от нежелательного воздействия пользователем, путем заблокирования необходимых компонентов SpeedButton (BitBtn, Button) до момента ввода необходимых данных. Так же ввели разрешение определенных символов в Edit и LabeledEdit. Код перевод курсора с названия на автора представлен в листинге 12.

Листинг12:

```
procedure TForm1.Frame21Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if not (key in [#13, #8,'a'..'я','.','A'..'Я','0'..'9',' ']) then key:=#0;
if key=#13 then Frame21.Edit2.SetFocus;
end;
```

Код проверки ввода текста представлен в листинге 13.

Листинг13:

```
procedure TForm1.Frame41LabeledEdit3Change(Sender: TObject);
begin
  if Frame41.LabeledEdit3.Text<>'' then Frame41.Button2.Enabled:=true;
  if Frame41.LabeledEdit3.Text='' then Frame41.Button2.Enabled:=false;
end;
```

При нажатии на иконку  , расположенную в верхней панели инструментов, либо нажать «Проект»/«Новый» открывается диалоговое сообщение (см. рис. 59). При нажатии на «Да» создается новый проект.

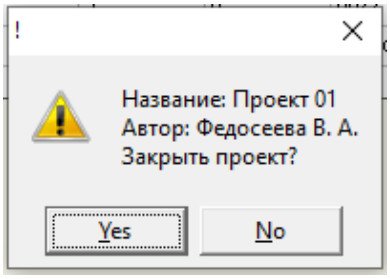


Рисунок 59 - Диалоговое окно

Код кнопки «новый проект» представлен в приложении С. Листинг остальных компонентов системы представлен в приложении Т.

В процессе реализации дипломной работы разработано руководство пользователю (приложение У).

Примеры созданных карт раскроя представлены в приложении Ф.

Таким образом, создана система раскроя ДО нестандартной формы на ГО тремя ранее описанными методами в среде программирования Delphy.

Вывод: реализована система автоматизации создания карт раскроя делового остатка на геометрические объекты. Реализованы все ее основные функции и задачи, с учетом основных ее ограничений и условий.

4 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

Эффективность любого специализированного программного обеспечения определяется его назначением, результатами использования по этому назначению, а также затратами на его создание и последующую эксплуатацию.

Целью экономической части является расчёт эффективности от внедрения автоматизированной системы учета техники в сервисном центре.

Основной задачей данного раздела является определение целесообразности разработки и величины экономического эффекта от внедрения программного продукта.

Внедрение системы учета техники в сервисном центре позволит получить следующие эффекты:

— Повышение качества и скорости обслуживания клиентов при приеме и выдаче неисправного оборудования.

— Рост производительности сотрудников при приеме и выдаче.

— Повышение прибыли предприятия за счет сокращения труда выполняемого человеком.

Для определения целесообразности разработки и величины экономического эффекта от внедрения программного продукта необходимо провести расчёт технико-экономических показателей.

4.1 Расчет трудоемкости разработки

Расчет трудоемкости разработки ПО – стоимость потраченного времени разработчиком на разработку ПО, определяется по формуле (43).

$$C=З \cdot Т \quad (43)$$

где З – среднедневная стоимость работ, денежные единицы, руб.;

Т – общая трудоемкость проекта, дни.

Стоимость работы одного часа составляет 150 рублей, таким образом, определили среднюю стоимость работ по формуле (44).

$$З=150 \cdot 8=1200 \text{ рублей в день} \quad (44)$$

Для разработки ПО потребовалось 40 рабочих дня, следовательно, стоимость трудоемкости составляет 48000 рублей.

$$С=З \cdot Т=40 \cdot 1200=48000 \text{ рублей} \quad (45)$$

4.2 Определение плановой себестоимости проведения работ

В себестоимость проведения работ включается стоимость оборудования и монтажа, определяется по формуле (46).

$$Ц=СО+М \quad (46)$$

Для установки и работы системы не требуется покупка нового оборудования. Поэтому, стоимость проведения работ равна нулю.

Ц=0 рублей.

4.3 Экономический эффект

Экономический эффект от использования программного продукта выражается в экономии затрат времени на обработку данных.

Рассчитав затраты на трудоемкость и себестоимость получим полную цену за работающую систему, определяемую по формуле (47).

$$Ц \text{ полн}=Ц+С=48000+0=48000 \text{ рублей} \quad (47)$$

Для расчета экономии возьмем стоимость работы одного часа сотрудника – 90 руб., среднее количество рабочих дней в месяце – 22 день.

Рассчитаем месячные расходы на оплату труда по формуле (48).

$$Змес=90 \cdot 8 \cdot 22=15840 \text{ руб} \quad (48)$$

Годовые расходы на оплату труда рассчитываются по формуле (49).

$$Згод=15840 \cdot 12=190080 \text{ руб.} \quad (49)$$

Без использования автоматизированной системы учета техники, время создания сохранной расписки и выдача оборудования одному клиенту составляет 7 минут. С использованием автоматизированной системы время составляет одну минуту. В течении дня сервисный центр обслуживает около 25 клиентов.

Найдем экономию времени на выполнение задач сотрудника.

Рассчитаем месячные расходы на оплату труда без использования АС по формуле (50).

$$Змес=90\cdot\frac{7}{60}\cdot25\cdot30=7875 \text{ руб} \quad (50)$$

Рассчитаем месячные расходы на оплату труда при использовании АС по формуле (51).

$$Змес=90\cdot\frac{1}{60}\cdot25\cdot30=1125 \text{ руб} \quad (51)$$

Итого рассчитаем ежемесячную экономиюпо формуле (52) и ежегодную экономию по формуле (53).

$$Эмес=7875-1125=6750 \text{ руб} \quad (52)$$

$$Эгод=6750\cdot12=81000 \text{ руб} \quad (53)$$

При стоимости системы 48000 руб. её установка окупится на восьмом месяце работы, а ежемесячная экономия составит 6750 руб.

Вывод: Произвели расчёт эффективности от внедрения автоматизированной системы учета техники в сервисном центре.При расчете эффективности выяснили, что себестоимость программного обеспечения по расчету стоимости трудоемкости составляет 48000 рублей. При эксплуатации программного обеспечения по рациональному размещению геометрических примитивов на деловом остатке нестандартной формы экономия производства составит 6750 руб. в месяц. При данной эффективности, стоимость программного обеспечения окупится на восьмом месяце экспуотации программы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения ВКР была разработана система рационального размещения графических примитивов на деловом остатке нестандартной формы.

Для достижения поставленной цели выполнено следующее:

— Выполнен аналитической обзор составления карт раскроя листового материала. Проанализированы известные модели формирования раскроя материала и методы их классификации;

— Разработана математическая модель рационального размещения графических примитивов на деловом остатке нестандартной формы. Выделены следующие методы создания карт раскроя делового остатка нестандартной формы: метод оптимизации «рюкзака», метод комбинаторики с применением жадного алгоритма, метод комбинаторики с применением полного перебора;

— На основе разработанной математической модели спроектировано и реализовано программное обеспечение рационального размещения графических примитивов на деловом остатке нестандартной формы в визуальной среде программирования Delphi;

— Выполнено экономическое обоснование.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Межгосударственный стандарт ГОСТ 30772-2001 "Ресурсосбережение. Обращение с отходами. Термины и определения" (введен в действие постановлением ГОССТАНДАРТА РФ от 28 декабря 2001 г. N 607-ст)

2. Р.А. Файзрахманов, Р.Т. Мурзакаев, В.С. Шилов, А.В. Буркова. Исследование бизнес-процесса учета делового остатка при раскрое листовых материалов // Вестник ПНИПУ электротехника, информационные технологии, системы управления № 7 – 2013

3. Файзрахманов Р.А., Братчиков И.А. Оптимизация процесса производства программного обеспечения на основе построения плана обучения исполнителей // Вестник Перм. Гос. Техн. Ун-та. Электротехника, информационные технологии, системы управления. – 2009. – № 3. – с. 74–80.

4. Чебышев П.Л. О кройке одежды. – Журн. Успехи матем. наук, 1946, 1,2.С.27.

5. Федоров Е.С. Симметрия и структура кристаллов. – М.:Изд-во АН СССР, 1949. – 411с.

6. Канторович Л.В. Математические методы в организации и планировании производства.– Л.:ЛГУ,1939.-60с.

7. Канторович Л.В. Методы рационального раскроя металла // Производственно - техн. бюллетень. - М.-1942.-35с.

8. Канторович Л. В., Горстко А. Б. Математическое оптимальное программирование.– М: Экономика,1968.-96с.

9. Канторович Л. В. ,Залгаллер В. А. Расчет рационального раскроя промышленных материалов.- Л.:Лениздат,1951.-199с.

10. Данциг Д.Б. Линейное программирование, его применение и обобщение. М.: Прогресс, 1966.-600с.

11. Ситдикова д.в. раскрой листового материала на заготовки // Казанский Национально Исследовательский Технический Университет им. А. Н. Туполева, г.Казань - УДК 691.5.

12. А. Ю. Васин в. Н. Задорожный решение производственной задачи одномерного раскроя материалов // омский государственный технический университет, омский научный вестник № 2 (110) 2012

13. А. Ф. Валеева, А. А. Петунин, Р. И. Файзрахманов // Применение конструктивной метаэвристики «муравьиная колония» к задаче гильотинного прямоугольного раскроя // Уфимский Государственный Авиационный Технический Университет

14. В. В. Мартынов, А. В. Бабель Метод регулярного размещения плоских геометрических объектов на базе геометрических преобразований // Уфа: УГАТУ, 2013

15. Р.Т. Мурзакаев, В.С. Шилов, А.В. Буркова Основные методы решения задачи фигурной нерегулярной укладки плоских деталей.

16. М. А. Верхотуров Задача нерегулярного раскроя фигурных заготовок: оптимизация размещения и пути режущего инструмента.

17. Верхотуров М. А., Верхотурова Г. Н., Задача нерегулярного размещения геометрических объектов: применение методов дискретной оптимизации // уфимский гос. Авиационный техн. Университет, кафедра математики и кибернетики.

18. П. В. Гуров Оптимизация методов фигурного раскроя листового материала. Аппроксимация методом касательных.

19. М. А. Чертов, Г. Е. Руденский, С. Г. Псахье Алгоритм группировки геометрических объектов при автоматическом раскрое листового материала с использованием локальных характеристик формы // институт физики прочности и материаловедения со ран, Томск, России

20. Н.А. Тюкачев Алгоритм определения принадлежности точки многоугольнику общего вида или многограннику с треугольными гранями // кафедра «программирование и информационные технологии», гоу впо «воронежский государственный университет»;

21. Верхотуров М. А., Верхотурова Г. Н., Брусиловский Д. П. Методы и алгоритмы нерегулярной двумерной упаковки объектов сложных геометрических форм. Рукописьдеп. Ввинити, №682-в97 от 05.03.97

22. Sykora A.M. Nesting problems: exact and heuristic algorithms. / A.M. Sykora // A Thesis for the degree of Doctor of Philosophy in the University of Valencia, Valencia, 2012.

23. Шапкин А.С., Мазаева Н.П. Математические методы и модели исследования операций Учебник. – 2-е изд., перераб. и доп. – М. Дашков и К, 2005. – 400 с

24. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. — 2-е изд. — М.: Вильямс, 2005. — 1296 с.

25. Короев Ю.И. Начертательная геометрия.– м. «кнорус». 2015. С.422. Isbn 978-5-406-04297-7

26. Richard Rhoad; George Milauskas; Robert Whipple. Geometry for Enjoyment and Challenge (неопр.). — new. — McDougal Littell (англ.)русск., 1991. — С. 717—718. — ISBN 0-86609-965-4.

27. Волков И.К., Загоруйко Е.А. Исследование операций (2000)

28. Кормен Т., Лейзер Ч. Алгоритмы. Построение и анализ.

29. Учебное пособие «Элементы комбинаторики», ГБПОУ КК «АМТ». 2016.

30. Р. Т. Мурзакаев, В. С. Шилов, А. В. Буркова // Основные методы решения задач фигурной нерегулярной укладки плоских деталей.

ПРИЛОЖЕНИЕ А

Блок схема алгоритма представления ГО i-ого типа в виде матрицы

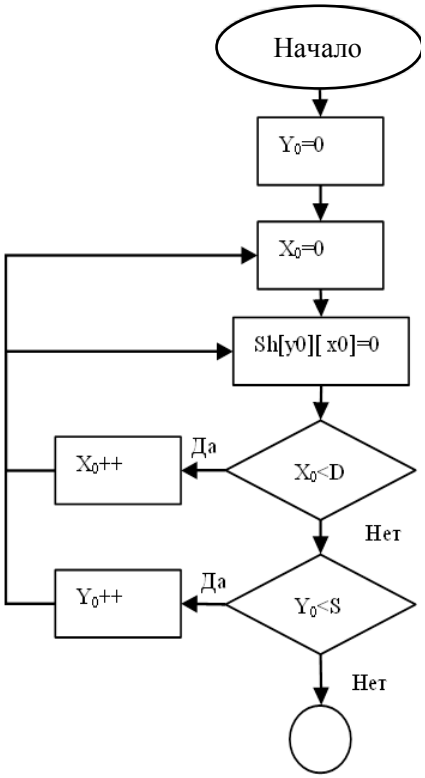


Рисунок А 1 - Блок схема алгоритма представления ГО i-ого типа в виде матрицы часть 1

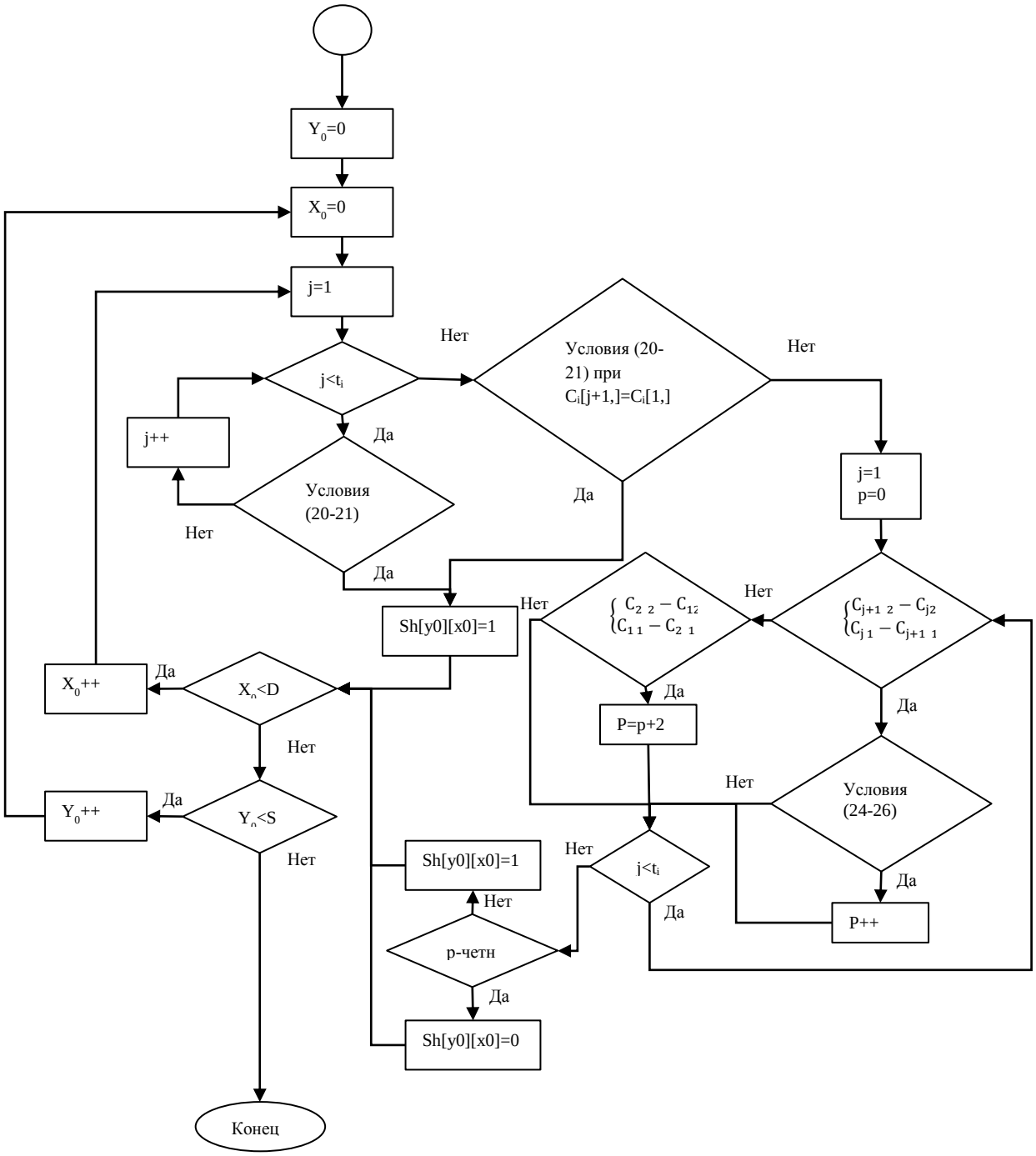


Рисунок А 2 - Блок схема алгоритма представления ГО i-ого типа в виде матрицы часть 2

ПРИЛОЖЕНИЕ Б

Схема работы системы

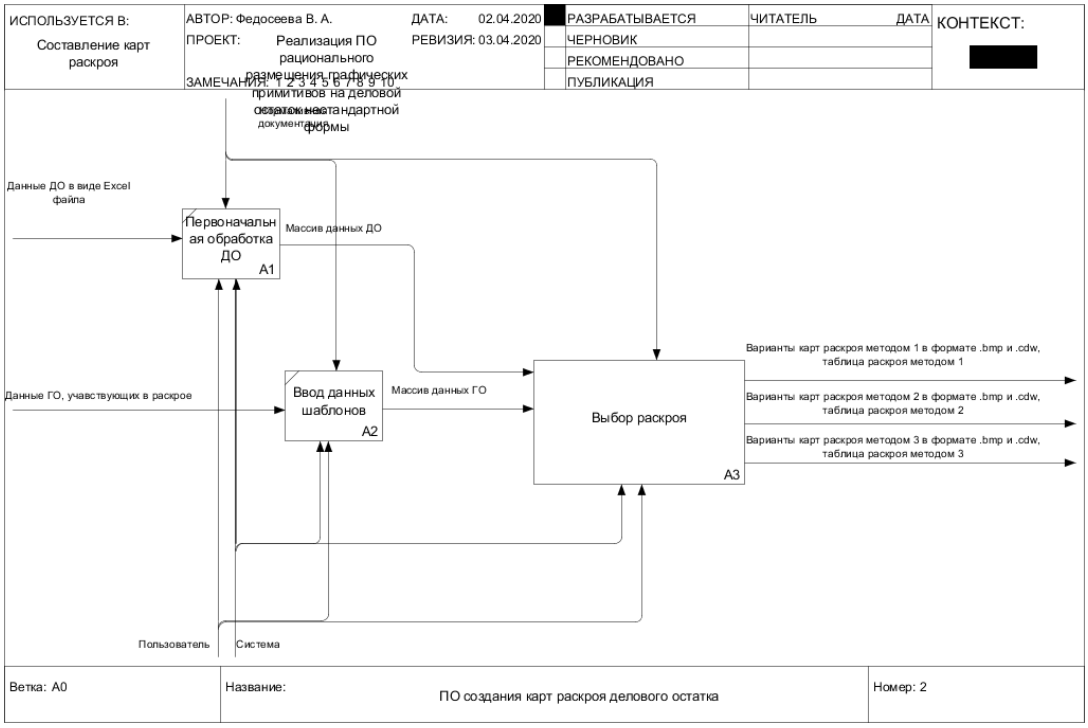


Рисунок Б 1 - Схема работы программы второго уровня

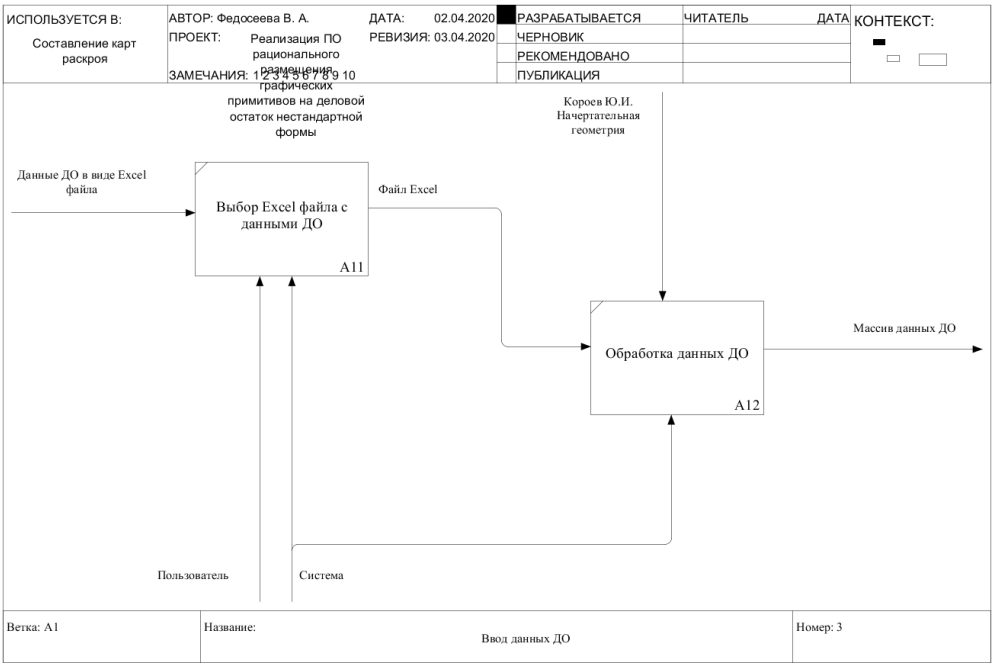


Рисунок Б 2 - Схема ввода данных ДО

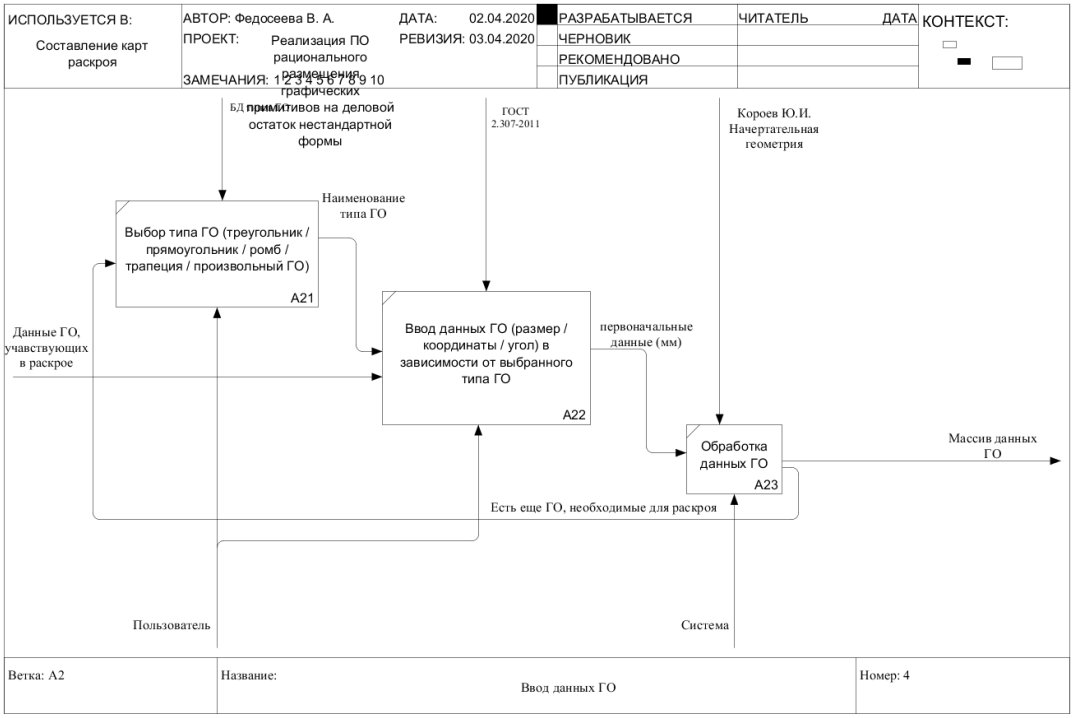


Рисунок Б 3 - Схема ввода данных ГО

ПРИЛОЖЕНИЕ В

Блок схема алгоритма работы системы

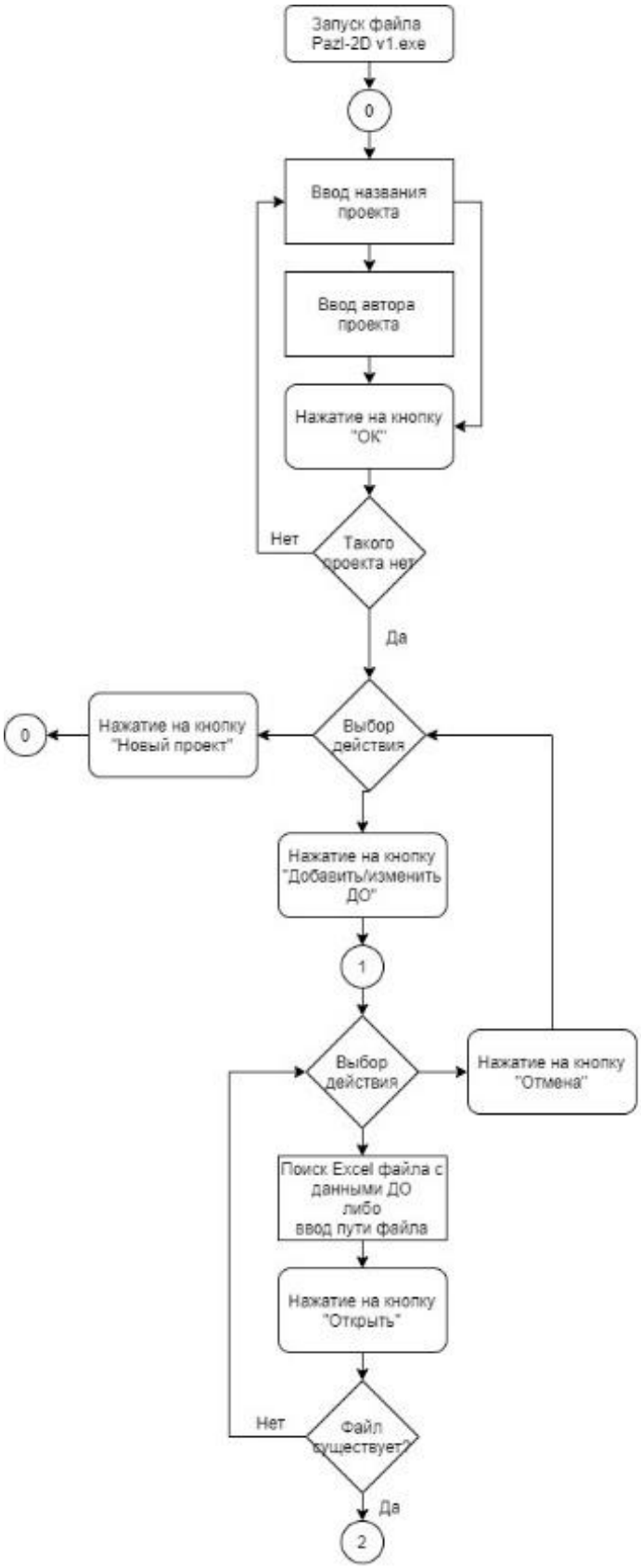


Рисунок В 1 - Блок схема алгоритма работы системы часть 1

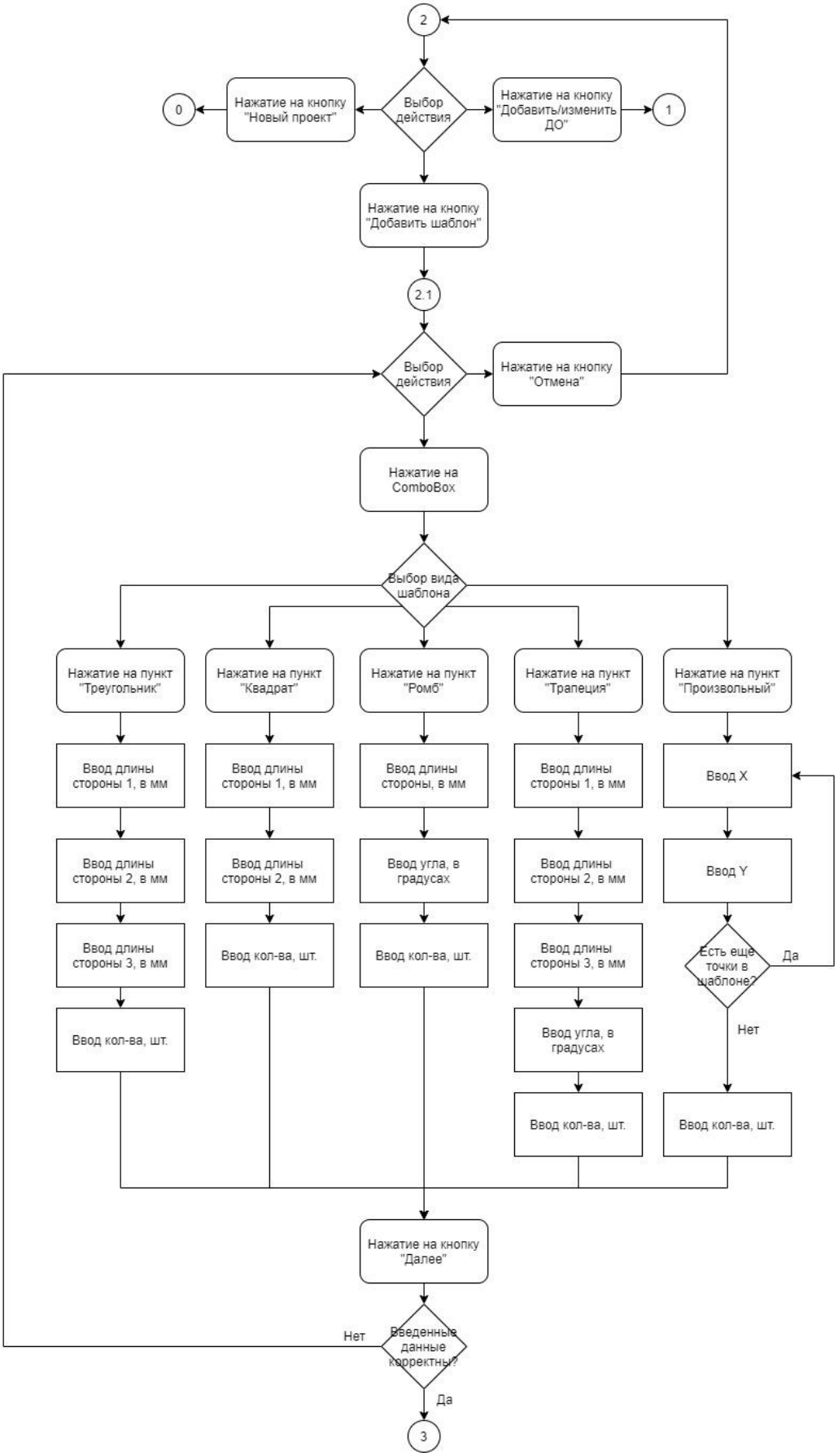


Рисунок В 2 - Блок схема алгоритма работы системы часть 2

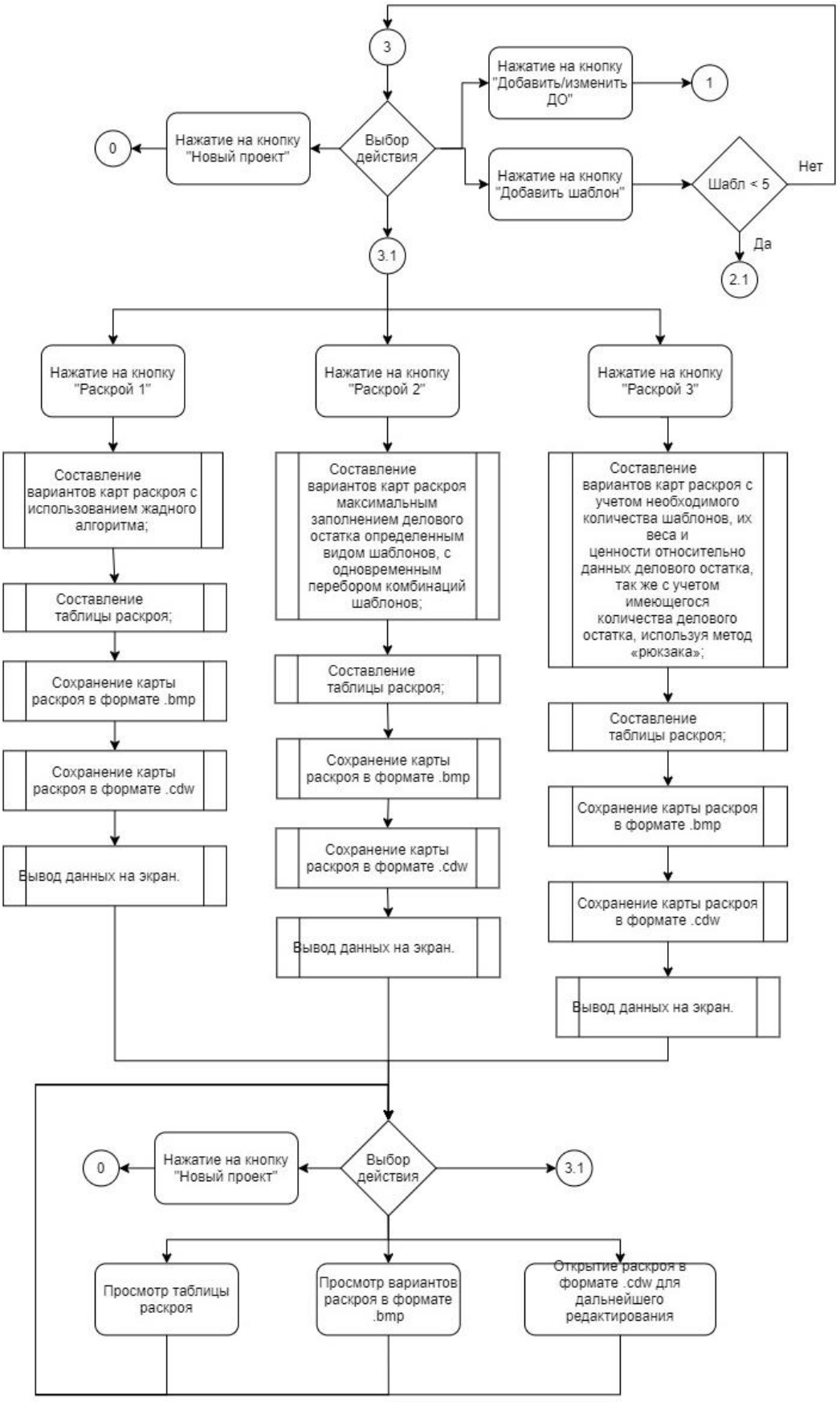


Рисунок В 3 - Блок схема алгоритма работы системы часть 3

ПРИЛОЖЕНИЕ Г

Листинг frame1

```
unitUnit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, jpeg, Buttons;

type
  TFrame2 = class(TFrame)
    Edit1: TEdit;
    Edit2: TEdit;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    BitBtn6: TBitBtn;
    StaticText29: TStaticText;
    StaticText27: TStaticText;
    StaticText26: TStaticText;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
implementation
{$R *.dfm}
end.
```

ПРИЛОЖЕНИЕ Д

Листинг Frame2 отображения входных и выходных данных

```
unit Unit3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, ExtCtrls, Grids, StdCtrls, ShellCtrls, DBCtrls,
  Buttons;
type
  TFrame3 = class(TFrame)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    TabSheet3: TTabSheet;
    StringGrid1: TStringGrid;
    Image1: TImage;
    Image2: TImage;
    StringGrid2: TStringGrid;
    StringGrid3: TStringGrid;
    Image3: TImage;
    LabeledEdit1: TLabeledEdit;
    TabSheet4: TTabSheet;
    StringGrid4: TStringGrid;
    Image4: TImage;
    CheckBox1: TCheckBox;
    TabSheet5: TTabSheet;
    StringGrid5: TStringGrid;
    Image5: TImage;
    ShellListView2: TShellListView;
    ShellListView3: TShellListView;
    ShellListView4: TShellListView;
    ShellListView5: TShellListView;
    Label1: TLabel;
    Image6: TImage;
    Image7: TImage;
    Splitter1: TSplitter;
    Image8: TImage;
    Splitter2: TSplitter;
    Splitter3: TSplitter;
    Image9: TImage;
    Image10: TImage;
    Splitter4: TSplitter;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    StaticText4: TStaticText;
    StaticText5: TStaticText;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
 // var StringGrid: TStringGrid;
implementation
{$R *.dfm}
end.
```

ПРИЛОЖЕНИЕ Е

Листинг компонентов добавления данных ГО

Е.1 Листинг Frame 4

```
unit Unit4;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Grids, Unit7, Unit6, Unit8, Unit9, Buttons;
type
  TFrame4 = class(TFrame)
    LabeledEdit1: TLabeledEdit;
    LabeledEdit2: TLabeledEdit;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    LabeledEdit3: TLabeledEdit;
    Button2: TButton;
    Image1: TImage;
    Button1: TButton;
    ComboBox1: TComboBox;
    Frame61: TFrame6;
    Frame71: TFrame7;
    Panel1: TPanel;
    Frame81: TFrame8;
    Frame91: TFrame9;
    BitBtn6: TBitBtn;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
implementation
{$R *.dfm}
end.
```

Е.2 Листинг Unit6

```
unit Unit6;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, jpeg;
type
  TFrame6 = class(TFrame)
    Image1: TImage;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;
implementation
{$R *.dfm}
end.
```

Е.3 Листинг Unit7

```
unit Unit7;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, jpeg;
type
  TFrame7 = class(TFrame)
    Image1: TImage;
    Edit3: TEdit;
    Edit1: TEdit;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
implementation
{$R *.dfm}
end.
```

Е.4 Листинг Unit8

```
unit Unit8;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, jpeg;
type
  TFrame8 = class(TFrame)
    Image1: TImage;
    Edit3: TEdit;
    Edit1: TEdit;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
implementation
{$R *.dfm}
end.
```

Е.5 Листинг Unit9

```
unit Unit9;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, jpeg;
type
  TFrame9 = class(TFrame)
    Image1: TImage;
    Edit3: TEdit;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit4: TEdit;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
implementation
{$R *.dfm}
end.
```

ПРИЛОЖЕНИЕ Ж

Код свойств компонентов визуализации этапов составления карт раскроя

```
object StaticText2: TStaticText
  Left = 213
  Top = 108
  Width = 12
  Height = 17
  Caption = '%'
  TabOrder = 2
end
object StaticText1: TStaticText
  Left = 101
  Top = 108
  Width = 69
  Height = 17
  Caption = Загрузилось
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  ParentFont = False
  TabOrder = 1
end
object ProgressBar1: TProgressBar
  Left = 100
  Top = 128
  Width = 781
  Height = 20
  Max = 1000
  TabOrder = 0
end
object Label5: TLabel
  Left = 185
  Top = 110
  Width = 6
  Height = 13
  Caption = '0'
end
object Label4: TLabel
  Left = 100
  Top = 172
  Width = 3
  Height = 13
end
object Label3: TLabel
  Left = 100
  Top = 153
  Width = 3
  Height = 13
end
```

```
object Label2: TLabel
  Left = 0
  Top = 13
  Width = 1004
  Height = 13
  Align = alTop
  Alignment = taCenter
Caption = Это может занять до 10 минут, не выключайте компьютер!
Layout = tlCenter
end
object Label1: TLabel
  Left = 0
  Top = 0
  Width = 1004
  Height = 13
  Align = alTop
  Alignment = taCenter
  BiDiMode = bdLeftToRight
  Caption = Идет раскрой делового остатка
  ParentBiDiMode = False
  Layout = tlCenter
end
```

ПРИЛОЖЕНИЕ 3

Фрагмент кода поиска уравнений прямых в фигуре ДО

```
for g:=1 to Trunc(data[0]*2) do
        if g mod 2<>0 then//еслих
begin
g1:=g1+1;
         if g=(n-2) then//если рассматривается последний X
begin
           Z[g1,1]:=Z[0,2]-Z[0,g+1];//a
           Z[g1,2]:=Z[0,g]-Z[0,1];   //b
           Z[g1,3]:=Z[0,g+1]*(Z[0,1]-Z[0,g])-Z[0,g]*(Z[0,2]-Z[0,g+1]) ;      //c
if Z[0,1]<Z[0,3] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                      else   if Z[0,1]>Z[0,3] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
           if Z[0,1]=Z[0,3] then
            if Z[0,1]<Z[0,5] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                      else   if Z[0,1]>Z[0,5] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
end
         else
         begin
           Z[g1,1]:=Z[0,g+3]-Z[0,g+1];//a
           Z[g1,2]:=Z[0,g]-Z[0,g+2] ;   //b
           Z[g1,3]:=Z[0,g+1]*(Z[0,g+2]-Z[0,g])-Z[0,g]*(Z[0,g+3]-Z[0,g+1]) ;      //c
if Z[0,g]<Z[0,g+2] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                      else  if Z[0,g]>Z[0,g+2] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
           if Z[0,g]=Z[0,g+2] then
            if Z[0,g]<Z[0,g+4] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                      else  if Z[0,g]>Z[0,g+4] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
end;
       end;
```

ПРИЛОЖЕНИЕ И

Фрагмент кода поиска матрицы фигуры ДО по найденным координатам

```
Xmin:=0;
Xmax:=data[4];
Ymin:=0;
Ymax:=data[5];
      V:=false;
      for y2:=Trunc(Ymin) to Trunc(Ymax) do
      begin
        g:=0;//сколько 1 встроке
        for x2:=Trunc(Xmin) to Trunc(Xmax) do
begin
         V:=false;
         //перебор прямых, посмотреть сколько пересечений,
         for g2:=1 to Trunc(data[0]) do  //проверка лежит ли точка на осях шаблона
begin
          j45:=Trunc(x2*Z[g2,1]+y2*Z[g2,2]+Z[g2,3]);
          if (j45=0)
          then
          begin
            if
((g2=Trunc(data[0]))and((y2>=Z[0,2])and(y2<=Z[0,g2*2])and(x2>=Z[0,1])and(x2<=Z[0,g2*2-
1])))
                then
                begin
                  Matr[y2][x2]:=1; V:=true;  break;
                end
                else
                  if ((Z[g2,1]=0)and(Z[g2,3]=0)) then   //еслипрямаяпараллельнаосиX
                  begin
                    if (Z[0,g2*2-1]>Z[0,g2*2+1])and(g2<>1)then
                      if ((x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1]))
                      then
                      begin
                        Matr[y2][x2]:=1; V:=true;  break;
                      end
                      else
                        if (Z[0,g2*2-1]<Z[0,g2*2+1])and(g2<>1)
                        then
                          if ((x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1]))
                          then
                          begin
                            Matr[y2][x2]:=1; V:=true;  break;
                          end;
                  end
                  else

if((Z[g2,1]<>0)and(Z[g2,3]<>0))and((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z[0,g2*2
-1])and(x2<=Z[0,g2*2+1]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-
1])and(x2>=Z[0,g2*2+1]))
                    then
```

```
                    begin
                      Matr[y2][x2]:=1; V:=true;  break;
                    end;
                  end;
                end;
                if (V=false) then
                begin
                  j:=0;
                  for g2:=1 to Trunc(data[0]) do
                  begin
                    if
(((Z[g2,1]<>0)and(Z[g2,2]<>0))or((Z[g2,1]<>0)and(Z[g2,2]=0)))and(((y2>=Z[0,g2*2])and(y2<
=Z[0,g2*2+2]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])))
                    then
                    begin
                      if              ((g2=Trunc(data[0]))and((Z[0,2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,1])<>0))     or     ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])<>0))or ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])=0))
                      then
                      begin
                        x1:=Trunc((Z[g2,2]*y2+Z[g2,3])/(-Z[g2,1]));
        if x1<=x2 then j:=j+1;//сколько точек пересечения до прямой x=0
        end;
                    end;
                  end;
                  if j mod 2<>0
        then Matr[y2][x2]:=1  //если не четное
                  else Matr[y2][x2]:=0;  //если четное
        end;
        end;//x
        end;//y
```

ПРИЛОЖЕНИЕК

Код взаимодействия Delphy с КОМПАС

К.1 Фрагмент кода закрытия компас

```
   if Kompas <> nil then
   begin
//принудительно закрыть
Kompas.Quit;
Kompas := nil;
end;
```

К.2 Фрагмент кода создания объекта автоматизации KOMPAS_Graphic

```
if Kompas = nil then
   begin
     {$IFDEF __LIGHT_VERSION__}
     Kompas:= KompasObject( CreateOleObject('KompasLT.Application.5') );
     {$ELSE}
     Kompas:= KompasObject( CreateOleObject('Kompas.Application.5') );
     {$ENDIF}
     if Kompas <> nil then
       Kompas.Visible := false;    //невиднопокомпастконаненужна
   end;
```

К.3 Фрагмент кода рисования ДО

```
   if Data[4]>Data[5] then n:=Trunc(500/Data[4]) else n:=Trunc(500/Data[5]);
   For i:=6 to (Trunc(Data[0])*2+5) do//переборкоординат
if i mod 2=0 then
     begin
       x:=Trunc(Data[i])*n+25;//том на 100 убрать тут чтоб видно было тк размер примера мал
Y:=Trunc(Data[i+1])*n+25;
        Frame31.Image3.Canvas.MoveTo(x,0);  //?? y
        Frame31.Image3.Canvas.LineTo(x,5);
        Frame31.Image3.Canvas.MoveTo(0,y);  //?? y
        Frame31.Image3.Canvas.LineTo(5,y);
        PointArray[j]:=Point(x,y); j:=j+1;
        //рисованиевкомпас
        if i<(Trunc(Data[0])*2+5-1) then Doc2.ksLineSeg( Trunc(Data[i]), Trunc(Data[i+1]), Trunc(Data[i+2]),  Trunc(Data[i+3]), 1 );
        if i=(Trunc(Data[0])*2+5-1) then Doc2.ksLineSeg( Trunc(Data[i]), Trunc(Data[i+1]), Trunc(Data[6]),  Trunc(Data[7]), 1 );
       end;
```

ПРИЛОЖЕНИЕ Л

Фрагмент кода сохранения координат и всех необходимых входных данных ГО относительно его типа

Л.1 Фрагмент кода сохранения данных при типе ГО «произвольный»

```
ifFrame41.ComboBox1.Text='Произвольный' then
begin
F[n][0]:=n0-1;//сохранение кол-ва точек в шаблоне
   //поиск всех необходимых данных
   //1. Поиск S фигуры    (ПОИСК ПЛОЩАДИ ФИГУРЫ  (по треугольникам с 1 до остальных точек))
//еслитреугольник
   if n0=4 then
   begin
     a:=(sqrt(sqr(F[n][5]-F[n][1])+sqr(F[n][6]-F[n][2])));
     b:=(sqrt(sqr(F[n][3]-F[n][1])+sqr(F[n][4]-F[n][2])));
     C:=(sqrt(sqr(F[n][5]-F[n][3])+sqr(F[n][6]-F[n][4])));
     x:=(c*c+b*b-a*a)/(2*b);
     h:=(sqrt(c*c-x*x));
DataF[n][1]:=h*b/2;
   end
   else  //если многоугольник
   begin
     //поиск площади по алгоритму шнурования
a:=0;
     b:=0;
     For i:=1 to (n0)*2 do
     begin
       if (i mod 2<>0)and(i<>(n0*2-1))
       then a:=a+F[n][i]*F[n][i+3]//еслих
       else if (i mod 2=0)and(i<>(n0*2)) then b:=b+F[n][i]*F[n][i+1]; //еслиу
     end;
     DataF[n][1]:=abs(a-b)/2;//0;
   end;
   Xmin:=F[n][1];
   Xmax:=F[n][1];
   Ymin:=F[n][2];
   Ymax:=F[n][2];
imin:=1; //только положение Y
imax:=2;
For i:=3 to (n0-1)*2 do
   begin
     // Поиск Xmin,Xmax
     if i mod 2<>0 then
begin
      if F[n][i]<=Xmin  then  Xmin:=F[n][i];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0 (если начинать с 0,0 то нормально)
if F[n][i]>Xmax then  Xmax:=F[n][i];
     end
     // Поиск Ymin,Ymax
     else
```

```
          begin
        if F[n][i]<=Ymin then begin Ymin:=F[n][i]; imin:=i; end;//НЕ СЧИТАЕТ,
СОХРАНЯЕТ ЗН.= 0       (если начинать с 0,0 то нормально)
        if F[n][i]>Ymax then begin Ymax:=F[n][i]; imax:=i; end;
        end;
          end;
          DataF[n][2]:=Xmin;//длина прямоугольника
          DataF[n][3]:=Xmax;
          DataF[n][4]:=Ymin;//ширина прмяоугольника
        DataF[n][5]:=Ymax;
          DataF[n][6]:=(Xmax-Xmin)*(Ymax-Ymin);
          DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
          DataF[n][8]:=5;
          Shabl:=Shabl+1;
          razr:=true;
         end;
```

Л.2 Фрагмент кода сохранения данных при типе ГО «Треугольник»

```
        if frame41.ComboBox1.Text='Треугольник' then
         begin
          a:=StrToFloat(frame41.Frame61.Edit1.Text);    //периметр
          b:=StrToFloat(frame41.Frame61.Edit2.Text);
          C:=StrToFloat(frame41.Frame61.Edit3.Text);
          if (a>b) and (a>c) then
          begin
            a1:=a; b1:=b; c1:=c;
          end
          else
          if (c>b) and (c>a)
          then
          begin
            a1:=c; b1:=b; c1:=a;
          end
          else
          if (b>a) and (b>c)
          then
          begin
            a1:=b; b1:=a; c1:=c;
          end
          else
          begin
            a1:=a; b1:=b; c1:=c;
        end;
          //проверка на корректность введенных данных: a+b>c, a+c>b, b+c>a, (a>0, b>0,
c>0)
        if
(((b1+c1)>a1)and((a1>9)and(a1<501))and((b1>9)and(b1<501))and((c1>9)and(c1<501))) then
        begin
            //сохранение координат
            F[n][0]:=3;
        F[n][1]:=0; //x1
```

```
        F[n][2]:=0; //y1
        x:=(a1*a1+b1*b1-c1*c1)/(2*a1);
        h:=sqrt(b1*b1-x*x);
        F[n][3]:=x; //x2
        F[n][4]:=h; //y2
        F[n][5]:=a1; //x3
        F[n][6]:=0; //y4
        DataF[n,1]:=a1*h/2;
        Xmin:=F[n][1];
        Xmax:=F[n][1];
        Ymin:=F[n][2];
      Ymax:=F[n][2];
        imin:=1; //только положение Y
      imax:=2;
        For i:=3 to 6 do
        begin
          // Поиск Xmin,Xmax
          if i mod 2<>0 then
      begin
            if F[n][i]<=Xmin then Xmin:=F[n][i];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
      if F[n][i]>Xmax then  Xmax:=F[n][i];
          end
          // Поиск Ymin,Ymax
          else
          begin
      if F[n][i]<=Ymin then begin Ymin:=F[n][i]; imin:=i;          end;//НЕ СЧИТАЕТ,
СОХРАНЯЕТ ЗН.= 0       (если начинать с 0,0 то нормально)
      if F[n][i]>Ymax then begin Ymax:=F[n][i]; imax:=i;    end;
      end;
        end;
        DataF[n][2]:=0;//длина прямоугольника
        DataF[n][3]:=Xmax;
        DataF[n][4]:=0;//ширина прмяоугольника
        DataF[n][5]:=Ymax;
        DataF[n][6]:=(Xmax-Xmin)*(Ymax-Ymin); {}
      DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
        DataF[n][8]:=1;
        Shabl:=Shabl+1;
        razr:=true;
      end;
      end;
```

Л.3 Фрагмент кода сохранения данных при типе ГО «прямоугольник»

```
      if frame41.ComboBox1.Text='Прямоугольник' then
        begin
          a:=StrToFloat(frame41.Frame71.Edit1.Text);    //периметр
          b:=StrToFloat(frame41.Frame71.Edit3.Text);
          if (((a<501)and(a>9))and((b<501)and(b>9))) then
      begin
        //сохранение координат
```

```
     F[n][0]:=4;
F[n][1]:=0; //x1
     F[n][2]:=0; //y1
     F[n][3]:=0; //x2
     F[n][4]:=b; //y2
     F[n][5]:=a; //x2
     F[n][6]:=b; //y2
     F[n][7]:=a; //x2
     F[n][8]:=0; //y2
     //матрицафигуры
     DataF[n,1]:=a*b; //площадь
     DataF[n][2]:=0;//длина
     DataF[n][3]:=a;
     DataF[n][4]:=0;//ширина
     DataF[n][5]:=b;
     DataF[n][6]:=a*b;
     DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
     DataF[n][8]:=2;
     Shabl:=Shabl+1;
     razr:=true;
   end;
end;
```

Л.4 Фрагмент кода сохранения данных при типе ГО «ромб»

```
if frame41.ComboBox1.Text='Ромб' then
  begin
    a:=StrToFloat(frame41.Frame81.Edit3.Text);    //сторона
    b:=StrToFloat(frame41.Frame81.Edit1.Text);   //уголбольший
    if (((b)>90)and(b<180)and((a<300)and(a>9))) then
begin
     //сохранение координат
     F[n][0]:=4;
a1:=cos((180-b)*3.14/180)*a;  //*3.14/180
     b1:=sin((180-b)*3.14/180)*a;
     F[n][1]:=0; //x1
     F[n][2]:=0; //y1
     F[n][3]:=a1; //x2
     F[n][4]:=b1; //y2
     F[n][5]:=a+a1; //x2
     F[n][6]:=b1; //y2
     F[n][7]:=a; //x2
     F[n][8]:=0; //y2
     DataF[n][2]:=0;//длина
     DataF[n][3]:=a+a1;
     DataF[n][4]:=0;//ширина
     DataF[n][5]:=b1;
     DataF[n][6]:=b1*(a+a1); {} //площадь
     DataF[n,1]:=a*b1; //площадьромба
     DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
     DataF[n][8]:=3;
     Shabl:=Shabl+1;
```

```
        razr:=true;
      end;
    end;
```

Л.5 Фрагмент кода сохранения данных при типе ГО «трапеция»

```
    if frame41.ComboBox1.Text='Трапеция' then
     begin
      a:=StrToFloat(frame41.Frame91.Edit3.Text);
      b:=StrToFloat(frame41.Frame91.Edit1.Text);
      c:=StrToFloat(frame41.Frame91.Edit2.Text);
      c1:=StrToFloat(frame41.Frame91.Edit4.Text)*3.14/180;
      if
((c1<90)and(c1>0)and(b<a)and((a<501)and(a>9))and((b<501)and(b>9))and((c<501)and(c>9)))
then
      begin
       F[n][0]:=4;
       F[n][1]:=0; //x1
       F[n][2]:=0; //y1
       F[n][3]:=a-c*cos(c1)-b; //x2
       F[n][4]:=sin(c1)*c; //y2
       F[n][5]:=a-c*cos(c1); //x2
       F[n][6]:=sin(c1)*c; //y2
       F[n][7]:=a; //x2
    F[n][8]:=0; //y2
       DataF[n][2]:=0;//длина прямоугольника
    DataF[n][3]:=a;
       DataF[n][4]:=0;//ширина
       DataF[n][5]:=sin(c1)*c;
       DataF[n][6]:=DataF[n][3]*DataF[n][5]; //площ
       DataF[n,1]:=(a+b)*(c*sin(c1))/2; //площадьтрапеции
       DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
       DataF[n][8]:=4;
       Shabl:=Shabl+1;
       razr:=true;
      end;
     end;
    end;
     //если больше 4 точек то можно дальше делать
    if (n0>=4) then Frame41.LabeledEdit3.Enabled:=true;
    end;
    end
    Else   //начало построение фигуры
    begin
      F[n][i]:=x;
      F[n][i+1]:=y;
      i:=i+2;
```

ПРИЛОЖЕНИЕ М

Листинг реализации карт раскроя методом №1

```
procedure TForm1.SpeedButton4Click(Sender: TObject);
var

k9,k10,k0,t6,j45,x1,y2,x2,j1,j,t1,k1,k2,g,g1,g2A,g2B,g3A,g3B,g2,g3,dl,sh,kol,g33,k,j2,i1,i2,i3,i
4,i5,imax,t,j3,a1,yg,j4,m:integer;
 Xmin,Xmax,Ymin,Ymax:real;
 c,h,x,S:real;
 Matr0 : array of array of integer;//матрицаДО
 a,b,t29,t30:double;
 max:double;
 Ksh: array [1..40,1..50] of integer;
 u,u1,u2,xu,yu,u3,u4,u5,rasp,Raskr3:integer;
 V,F11,razrehenie:Boolean;
 Z: array [0..100,0..100] of real;  //массивскоординатамифигуры
 F1: array [1..15] of real;
 RasterFormatParam:ksRasterFormatParam;
 p:Integer;
begin
 p:=Form1.Width-200;
 Frame101.ProgressBar1.Width:=p;
 // Создатьобъектавтоматизации KOMPAS_Graphic
 Panel1.Enabled:=false;//заблокированныпанелькнопок
 Frame31.PageControl1.ActivePageIndex:=3;
 Frame31.StringGrid4.ColCount:=4+Shabl;
 if Shabl<=2 then Frame31.StringGrid4.Width:=(4+Shabl)*65+10
 else  Frame31.StringGrid4.Width:=(4+2)*65+10;
 Frame31.StringGrid4.Cells[0, 0]:='№';
 Frame31.StringGrid4.Cells[1, 0]:='ДО';
 for i:=1 to Shabl do Frame31.StringGrid4.Cells[i+1, 0]:=IntToStr(i);   //видшаблона,
использовавшегопираскрое;
 Frame31.StringGrid4.Cells[Shabl+2, 0]:='S ост.';  //площальостатка
 Frame31.StringGrid4.Cells[Shabl+3, 0]:='% заполнения';  //площальостатка
 Frame31.Visible:=false;
 Frame31.StaticText4.Visible:=false;
 Frame31.ShellListView5.Visible:=true;
 Frame101.Visible:=true;
 Frame101.ProgressBar1.Visible:=true;
 Frame101.ProgressBar1.Position:=0;
 Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
 Frame101.Label5.Refresh;
 Frame101.Refresh;
 t29:=0;
 for j:=1 to Shabl do
 t29:=t29+(DataF[j][7]);
 t29:=950/t29;
 t30:=0;
 Frame101.Label3.Caption:=''; Frame101.Label3.Refresh;
 Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
 Frame101.Label3.Caption:='Созданиеобъектаавтоматизации KOMPAS_Graphic';
```

```
  Frame101.Label3.Refresh;
  Frame101.Label4.Caption:=' ';
 Frame101.Label4.Refresh;
  //рисование   в КОМПАС
  // Создать объект автоматизации KOMPAS_Graphic
  if Kompas = nil then
 begin
    {$IFDEF __LIGHT_VERSION__}
    Kompas:= KompasObject( CreateOleObject('KompasLT.Application.5') );
    {$ELSE}
    Kompas:= KompasObject( CreateOleObject('Kompas.Application.5') );
    {$ENDIF}
    if Kompas <> nil then
      Kompas.Visible := false;    //невиднопокомпастконаненужна
 end;
  //работа с листом
  SetLength(Matr0,(Trunc(Data[5])+1000),(Trunc(Data[4])+1000));//выделение области
памяти для матрицы
 ForceDirectories(ProgDIR       +'\show       projects'+'\'+Frame21.edit1.text+'\'+'Nesting
№3');//созданиепапкисименем
   m:=0;
   for j:=1 to 5 do
   begin
     Ksh[1,j]:=0;//видшаблона
     Ksh[3,j]:=0;//Wi = S ВЕС
 Ksh[2,j]:=0;
     Ksh[4,j]:=0; //кол-во
 end;
   for j:=1 to Shabl do
   begin
     Ksh[1,j]:=j;//видшаблона
     Ksh[3,j]:=Trunc(DataF[j,1]);//Wi = S ВЕС
     if DataF[j][3]>DataF[j][5] then Ksh[2,j]:=Trunc((DataF[j,1]/DataF[j][3]))//Ui = max/S
СТОИМОСТЬ
                          else Ksh[2,j]:=Trunc((DataF[j,1]/DataF[j][5]));//Ui = max/S
     Ksh[4,j]:=Trunc(DataF[j][7]); //кол-во
     m:=m+Trunc(DataF[j][7]);  { }
     //ShowMessage('1:'+FloatToStr(Ksh[1,j])+'                    2:'+FloatToStr(Ksh[2,j])+'
3:'+FloatToStr(Ksh[3,j])+'   4:'+FloatToStr(Ksh[4,j])+'   5:'+FloatToStr(Ksh[5,j]));
 end;//К для данного ДО
   //ShowMessage('всего необходимо раскроить: '+FloatToStr(m)+' шаблонов');
   //ShowMessage('стоимость    первоначальная:          шб1:'+FloatToStr(Ksh[2,1])+'
шб2:'+FloatToStr(Ksh[2,2])+'    шб3:'+FloatToStr(Ksh[2,3])+'    шб4:'+FloatToStr(Ksh[2,4])+'
шб5:'+FloatToStr(Ksh[2,5]));
   //сортировка  Ksh[i,j] в зависимости от  Ui = max/S  СТОИМОСТИ
   {сортировка данных Пузырьковая сортировка в Delphi.}
 if Shabl>1 then
   begin
   for i:=1 to Shabl do
     for j:=1 to Shabl-i do
       if Ksh[2,j]<Ksh[2,j+1] then
       begin {Обменэлементов}
```

```
              j3:=Ksh[2,j];
              Ksh[2,j]:=Ksh[2,j+1];
              Ksh[2,j+1]:=j3;

              j3:=Ksh[1,j];
              Ksh[1,j]:=Ksh[1,j+1];
              Ksh[1,j+1]:=j3;

              j3:=Ksh[3,j];
              Ksh[3,j]:=Ksh[3,j+1];
              Ksh[3,j+1]:=j3;

              j3:=Ksh[4,j];
              Ksh[4,j]:=Ksh[4,j+1];
              Ksh[4,j+1]:=j3;
            end;
          end;
          //ShowMessage('стоимостьОТСОРТИРОВАННАЯ:      шб1:'+FloatToStr(Ksh[2,1])+'
шб2:'+FloatToStr(Ksh[2,2])+'    шб3:'+FloatToStr(Ksh[2,3])+'    шб4:'+FloatToStr(Ksh[2,4])+'
шб5:'+FloatToStr(Ksh[2,5]));
        j3:=0;
          j4:=0;
          //сколько всего шаблонов раскраиваться будет
        for i1:=0 to Ksh[4,1] do
           for i2:=0 to Ksh[4,2] do
             for i3:=0 to Ksh[4,3] do
               for i4:=0 to Ksh[4,4] do
                 for i5:=0 to Ksh[4,5] do
                   begin
        j3:=j3+1; //сколько МАКСИМУМ вариантов раскроя для освободжение памяти для
STabl2
        j4:=j4+i1+i2+i3+i4+i5;//сколькомаксимумдля Raskroi2
        end;
          //освобождение матрицы
          SetLength(STabl3,(100),(j3));//выделение области памяти для матрицы      (у или
строка макс, х или столбец максимально)
        SetLength(Raskroi3,(j4+100),(100));
          k:=0;
          rs:=1;
        t6:=1;
          repeat   //пока все необходимые шаблоны небудут распределены на ДО
            //матрица ДО первоначального вида
            For j:=0 to (Trunc(Data[5])) do
        For j1:=0 to (Trunc(Data[4])) do
              Matr0[j][j1]:=Matr[j][j1];
            //переборшаблонов
            for i:=1 to (Shabl) do
            begin
              k0:=0;
              Frame101.Label3.Caption:=''; Frame101.Label3.Refresh;
              Frame101.Label3.Caption:='Вмещениемаксимальногокол-вашаблона
'+FloatToStr(Trunc(Ksh[1,i]))+' вида.';
```

```
        Frame101.Label3.Refresh;
        Frame101.Label4.Caption:='';
      Frame101.Label4.Refresh;
        //перебор матрицы для вставки шаблона
      For k1:=(0) to Trunc(Data[5]) do//Y
                  begin
       if Ksh[4,i]=0 then  break; //если все вмешено то прерывание
                  for k2:=(0) to Trunc(Data[4]) do //X
       begin
        Frame101.ProgressBar1.Position:=Trunc((t29/(Data[5]*Data[4]))+t30);
        t30:=(t29/(Data[5]*Data[4]))+t30;
        Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
      Frame101.Label5.Refresh;
        if Ksh[4,i]=0 then  break; //если все вмешено то прерывание
        //проверка войдет ли в до
      //обнулениематриц  Z[g,0...100]  for g:=0 to 100 do
        For t:=0 to 50 do
         For t1:=0 to 50 do
          Z[t,t1]:=0;
        ////////////////////////////////////////////////////////////////////////////////////////////
        if Matr0[k1][k2]=1 then
        begin
         a1:=360;
         f11:=False;
         j:=Ksh[1,i];
         repeat
          razrehenie:=true;
          if ((F11=true)) then
          begin
           g1:=1;
           //отражениефигуры
           for g:=Trunc(F[j,0]*2) downto 1 do
            if g mod 2=0 then  F1[g1+1]:=F[j,g]        //если y
            else
            begin
             F1[g1]:=F[j,g]-2*F[j,g]+DataF[j][3];
      g1:=g1+2;
              //вывод координат для проверки
      //ShowMessage('1. координаты: '+FloatToStr(Z[0,g])+';'+FloatToStr(Z[0,g+1]));
      end;  //если x
              //сохранили коорлинаты нового положения фигуры
      for g:=1 to Trunc(F[j,0]*2) do
               ifgmod 2<>0 then//еслих
      begin//по математическим правилам до ближайшего целого:
      Z[0,g]:=Round(F1[g]*cos(a1*3.14/180)-F1[g+1]*sin(a1*3.14/180));//x   //k2
                Z[0,g+1]:=Round(F1[g+1]*cos(a1*3.14/180)+F1[g]*sin(a1*3.14/180));//y
//k1
      //добавить к координатам текущее положение
                //если хотя бы одна новая координата не входит в ДО то прерывания
поиска
      //разрешение:=true;
                //((g>=0) and (g1>=0))and((g<=Data[5]) and (g1<=Data[4]))
```

```
                    k9:=Round(Z[0,g]+k2); //x
                    k10:=Round(Z[0,g+1]+k1); //y
                    if NOT ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and (k9<=Data[4]))))
then begin razrehenie:=false; break; end;
                    if ((((k9>0) and (k10>0))and((k10<Data[5]) and (k9<Data[4])))) then
                    if Matr0[k10][k9]=0 then begin razrehenie:=false; break; end;
                    //ShowMessage('1.                                    координаты:
'+FloatToStr(Z[0,g])+';'+FloatToStr(Z[0,g+1]));
        end;
                end
                else//сохранили коорлинаты нового положения фигуры без
отражения//сохранили коорлинаты нового положения фигуры
        for g:=1 to Trunc(F[j,0]*2) do
                  if g mod 2<>0 then//еслих
        begin
        //// по математическим правилам до ближайшего целого:
        Z[0,g]:=Round(F[j,g]*cos(a1*3.14/180)-F[j,g+1]*sin(a1*3.14/180));//x
                    Z[0,g+1]:=Round(F[j,g+1]*cos(a1*3.14/180)+F[j,g]*sin(a1*3.14/180));//y
        //добавить к координатам текущее положение
                    //если хотя бы одна новая координата не входит в ДО то прерывания
поиска
        //разрешение:=true;
                    k9:=Round(Z[0,g]+k2); //x
                    k10:=Round(Z[0,g+1]+k1); //y
                    if NOT ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and (k9<=Data[4]))))
then begin razrehenie:=false; break; end;
                    if ((((k9>0) and (k10>0))and((k10<Data[5]) and (k9<Data[4])))) then
                    if Matr0[k10][k9]=0 then begin razrehenie:=false; break; end;
                    //ShowMessage('1.                                    координаты:
'+FloatToStr(Z[0,g])+';'+FloatToStr(Z[0,g+1]));
                  end;
                if razrehenie=true then
                begin
        g1:=0;//прямых
                  //заполнение уравнений прямых в фигуре
        for g:=1 to Trunc(F[j,0]*2) do
                  if g mod 2<>0 then//еслих
                  begin
                    g1:=g1+1;
                    if g=F[j,0]*2-1 then
                    begin
                      Z[g1,1]:=Z[0,2]-Z[0,g+1] ;//a
                      Z[g1,2]:=Z[0,g]-Z[0,1] ;   //b
                      Z[g1,3]:=Z[0,g+1]*(Z[0,1]-Z[0,g])-Z[0,g]*(Z[0,2]-Z[0,g+1]) ;      //c
        if Z[0,1]<Z[0,3] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                      else  if Z[0,1]>Z[0,3] then Z[g1,0]:=-1; //знак (в какую сторону вектор)
                      if Z[0,1]=Z[0,3] then
                        if Z[0,1]<Z[0,5] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                        else  if Z[0,1]>Z[0,5] then Z[g1,0]:=-1; //знак (в какую сторону вектор)
        end
                    else
                    begin
```

```
                    Z[g1,1]:=Z[0,g+3]-Z[0,g+1] ;//a
                    Z[g1,2]:=Z[0,g]-Z[0,g+2] ;   //b
                    Z[g1,3]:=Z[0,g+1]*(Z[0,g+2]-Z[0,g])-Z[0,g]*(Z[0,g+3]-Z[0,g+1]) ;      //c
      if Z[0,g]<Z[0,g+2] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                    else  if Z[0,g]>Z[0,g+2] then Z[g1,0]:=-1; //знак (в какую сторону вектор)
                    if Z[0,g]=Z[0,g+2] then
                      if Z[0,g]<Z[0,g+4] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                      else   if Z[0,g]>Z[0,g+4] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
      end;
                   //ShowMessage('прямая-'+FloatToStr(g1)+'          a='+FloatToStr(Z[g1,1])+'
b='+FloatToStr(Z[g1,2])+' c='+FloatToStr(Z[g1,3]));
      end;
                //поиск минимума максимума новых координат фигуры
                Xmin:=Z[0][1];
                Xmax:=Z[0][1];
                Ymin:=Z[0][2];
      Ymax:=Z[0][2];
                For g:=3 to Trunc(F[j,0]*2) do
                begin
                  if g mod 2<>0 then
                  begin
      if Z[0][g]<=Xmin then Xmin:=Z[0][g];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
      if Z[0][g]>Xmax then  Xmax:=Z[0][g];
                  end
                  // Поиск Ymin,Ymax
                  else
                  begin
      if Z[0][g]<=Ymin then Ymin:=Z[0][g];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
      if Z[0][g]>Ymax then  Ymax:=Z[0][g];
                  end;
                end;
                //ShowMessage('Xmin='+FloatToStr(Xmin)+'      Xmax='+FloatToStr(Xmax)+'
Ymin='+FloatToStr(Ymin)+' Ymax='+FloatToStr(Ymax));
      //поиск матрицы фигуры по найденным координатам
                //ShowMessage('поиск матрицы фигуры по найденным координатам');
      //онулениематрицышаблона
                For y2:=Trunc(Ymin) to Trunc(Ymax) do
                  For x2:=Trunc(Xmin) to Trunc(Xmax) do
      MatrSH[y2][x2]:=0;
                //поиск матрицы фигуры по найденным координатам
                //сохраняем матрицу шаблона в MatrSH[y2][x2]
      V:=false;
                for y2:=Trunc(Ymin) to Trunc(Ymax) do
                begin
                  g:=0;//сколько 1 встроке
                  for x2:=Trunc(Xmin) to Trunc(Xmax) do
      begin
                    V:=false;
                    //перебор прямых, посмотреть сколько пересечений,
```

```
                    for g2:=1 to Trunc(F[j,0]) do   //проверка лежит ли точка на осях шаблона
        begin
                     j45:=Trunc(x2*Z[g2,1]+y2*Z[g2,2]+Z[g2,3]);
                    //ShowMessage('точка  '+FloatToStr(x2)+'  ,  '+FloatToStr(y2)+'прямая
'+FloatToStr(g2)+'>>>>>должнобыть>=0       >>>>>>>>>       '+FloatToStr(j45));
//+FloatToStr(x2)+'*'+FloatToStr(Z[g2,1])+'+'+FloatToStr(y2)+'*'+FloatToStr(Z[g2,2])+'+'+Flo
atToStr(Z[g2,3])+'='+FloatToStr(j45));
                    if                                                          (j45=0)//and
(((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1])))
//or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1]))
                    then
                    begin
                     if
((g2=Trunc(F[j,0]))and((y2>=Z[0,2])and(y2<=Z[0,g2*2])and(x2>=Z[0,1])and(x2<=Z[0,g2*2-
1]))) then
                      begin
                       //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')       >>>>>>>
'+'конечнаяпрямая');
                       MatrSH[y2][x2]:=1;
                       V:=true;
                       break;
                      end
                      else
                       if ((Z[g2,1]=0)and(Z[g2,3]=0)) then   //еслипрямаяпараллельнаосиX
                       begin
                        if (Z[0,g2*2-1]>Z[0,g2*2+1])and(g2<>1)then
                         if ((x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1])) then
                         begin
                          //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')     >>>>>>>
'+'прямаяПАРАЛперевернутая');
                           MatrSH[y2][x2]:=1; V:=true;  break;
                         end
                         else if (Z[0,g2*2-1]<Z[0,g2*2+1])and(g2<>1)then if ((x2>=Z[0,g2*2-
1])and(x2<=Z[0,g2*2+1])) then
                           begin
                            //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')   >>>>>>>
'+'прямаяПАРАЛнормальная');
                             MatrSH[y2][x2]:=1;
                             V:=true;
        break;
                           end;
                       end
                       else //если не паралельная и не конечная то
        if((Z[g2,1]<>0)and(Z[g2,3]<>0))and((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z
[0,g2*2-
1])and(x2<=Z[0,g2*2+1]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-
1])and(x2>=Z[0,g2*2+1]))
                        then
                        begin
                         //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')       >>>>>>>
'+'простопрямая');
                         MatrSH[y2][x2]:=1;
```

```
                        V:=true;
                        break;
                      end;
                  end;
                end;
                if (V=false) then
                begin
                  j1:=0;
                  SetLength(coord,Trunc(F[j,0])+100);//выделениепамяти
                  //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')          >>>>>>>
'+'точкавнепаралибоковых');
      for g2:=1 to Trunc(F[j,0]) do  //проверка лежит ли точка на осях шаблона
      begin
                   if
(((Z[g2,1]<>0)and(Z[g2,2]<>0))or((Z[g2,1]<>0)and(Z[g2,2]=0)))and(((y2>=Z[0,g2*2])and(y2<
=Z[0,g2*2+2]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2]))) then  //
                   begin
                     if      ((g2=Trunc(F[j,0]))and((Z[0,2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,1])<>0))    or    ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])<>0))or ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])=0))
                     then
                     begin
                       x1:=Trunc((Z[g2,2]*y2+Z[g2,3])/(-Z[g2,1]));
                       //ShowMessage('(T.    перес)x1='+FloatToStr(x1)+'          (Расм.
T.)x2='+FloatToStr(x2)+'   >>>>>>> if x1<=x2 then j:=j+1;');
                       if x1<=x2 then
                         begin
                           if j1=0 then
      begin
                             j1:=j1+1;//сколько точек пересечения до прямой x=0
      coord[j1]:=x1;
                           end
                           else
                           begin
                             cr1:=true;
                             for cr:=1 to j1 do
                               if coord[cr]=x1 then begin cr1:=false; break; end;
      if cr1=true then
                             begin
                               j1:=j1+1;//сколько точек пересечения до прямой x=0
      coord[j1]:=x1;
                             end;
                           end;
                         end;
                     end;
                   end;
                 end;
               end;
               if j1 mod 2<>0 then MatrSH[y2][x2]:=1  //еслинечетное
               else MatrSH[y2][x2]:=0;  //есличетное
               //ShowMessage('пересекаетдоточких='+FloatToStr(x2)+'
'+FloatToStr(j)+' раз');
```

```
                    end;
                    //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')          >>>>>>>
'+FloatToStr(Matr[y2][x2]));
        end;  //x
                end;   //y
                //наложение матрицы фигуры на матрицу ДО
                //ShowMessage('наложение матрицы фигуры на матрицу ДО');
        if
((Xmin+k2)>=0)and((Xmax+k2)<=Trunc(Data[4]))and((Ymin+k1)>=0)and((Ymax+k1)<=Trun
c(Data[5]))
                then
                begin
                  //ShowMessage('проверяемфигуру, вдиапазонвходит');
                  V:=true;
                  for g:=Trunc(Ymin) to Trunc(Ymax) do
                  begin
                    for g1:=Trunc(Xmin) to Trunc(Xmax) do
                      if MatrSH[g][g1]=1 then  if MatrSH[g][g1]<>Matr0[g+k1][g1+k2] then
begin V:=false;  break; end;     //
                     if V=false then break;
                  end;
                end
                else V:=false;
                //ShowMessage('V:= '+BoolToStr(V));
                if V=true then
                begin
                  //ShowMessage('онудениематрицыеслифигуравходит');
                  for g:=Trunc(Ymin) to Trunc(Ymax) do
                    for g1:=Trunc(Xmin) to Trunc(Xmax) do
                      if MatrSH[g][g1]=1 then Matr0[g+k1][g1+k2]:=0;
                   Matr0[k1][k2]:=1;
                   //ShowMessage('вписали '+FloatToStr(k)+'; осталось '+FloatToStr(L[t]));
                   //сохранениекоординат
                   Frame101.ProgressBar1.Position:=Trunc(t29+t30);
                   t30:=t29+t30;
                   k:=k+1;
                   k0:=k0+1;
                   Ksh[4,i]:=Ksh[4,i]-1;
                   //ShowMessage('вписали          '+FloatToStr(k)+';          осталось
'+FloatToStr(Ksh[4,i]));
        Raskroi3[k+rs-1,2]:=Ksh[1,i];//сохранение номера шаблона
        Raskroi3[k+rs-1,3]:=k2;//x0
                   Raskroi3[k+rs-1,4]:=k1;//y0
        Raskroi3[k+rs-1,5]:=a1;//ориентация(угол поворота относительно x0y0 против
часовой стрелки
                   STabl3[1,t6]:=1;  //перебор № ДО
                   STabl3[2,t6]:=t6;  //перебор № листа
        STabl3[i+2,t6]:=k0;
                   Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
                   Frame101.Label4.Caption:='Вместили '+FloatToStr(k)+' шт.';
                   Frame101.Label4.Refresh;
                   break;//прерываниепоискаугла
```

```
                    end;
          end;//КОНЕЦ действие если разрешение:=true;
          if  DataF[j][8]<>2 then
                    if
((Frame31.LabeledEdit1.Text<>'')and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                   else yg:=90;
                   if DataF[j][8]=2 then
                   begin
                     if (DataF[j][3]<>DataF[j][5]) then//еслипрямоугольник
                     begin
                       if                Frame31.CheckBox1.Checked                =                false
then//есликлаишазапретаненажатато
                         if
((Frame31.LabeledEdit1.Text<>'')and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                         else yg:=90;
                       if Frame31.CheckBox1.Checked = true then yg:=90;
                     end;
                    if (DataF[j][3]=DataF[j][5]) then//если rdf
                    begin
                       if                Frame31.CheckBox1.Checked                =                false
then//есликлаишазапретаненажатато
                         if
((Frame31.LabeledEdit1.Text<>'')and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                         else yg:=360;
                       if Frame31.CheckBox1.Checked = true then yg:=360;
                    end;
                   end;

Frame101.ProgressBar1.Position:=Trunc((t29/(Data[5]*Data[4]*(360/yg)))+t30);
                   t30:=(t29/(Data[5]*Data[4]*(360/yg)))+t30;
                   Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
                   Frame101.Label5.Refresh;
                   a1:=a1-yg;
                   //отображение
                   //if ((a1=0)and(F11=false)and(DataF[n][8]<>2)) then begin a1:=360; F11:=true;
end;  //  поворотфигуры
                 until  ((a1=0));
                 if ((V=true)) then Matr0[k1][k2]:=0;
               end;//если =1
             end;
           end;
         end;
         //сохранение
         //ShowMessage('раскрой'+FloatToStr(i)+' всегошаблонов'+FloatToStr(k));
         Raskroi3[rs,1]:=k;
```

```
            STabl3[8,t6]:=Data[2]-
(DataF[1][1]*STabl3[3,t6]+DataF[2][1]*STabl3[4,t6]+DataF[3][1]*STabl3[5,t6]+DataF[4][1]*S
Tabl3[6,t6]+DataF[5][1]*STabl3[7,t6]);//остатокотраскроя (поПРЯМОУГОЛЬНИКАМ)
        // Создать объект автоматизации KOMPAS_Graphic  ранее
            //Получаем и заполняем интерфейс параметров документа
        DocumentParam:=ksDocumentParam(kompas.GetParamStruct(ko_DocumentParam));
            DocumentParam.Init();
            DocumentParam.type_:=ksDocumentFragment;//Drawing;
            //Получаеминтерфейсдокумента
            Doc2:=ksDocument2D(kompas.Document2D);
            //Создаемновыйчертеж
            Doc2.ksCreateDocument(DocumentParam);
            //заполнениетаблицы
            Frame31.StringGrid4.Cells[0, t6]:=FloatToStr(STabl3[2,t6]);//номервариантараскроя
            Frame31.StringGrid4.Cells[1,  t6]:=FloatToStr(STabl3[1,t6]);  //видделовогоостатка,
использовавшегопираскрое
            for  i1:=1  to  Shabl  do  Frame31.StringGrid4.Cells[i1+1,
t6]:=FloatToStr(STabl3[i1+2,t6]);  //видшаблона,  использовавшегопираскрое
//видшаблона, использовавшегопираскрое;
            Frame31.StringGrid4.Cells[Shabl+2,  t6]:=FloatToStr(Trunc(STabl3[8,t6]));
//площальостатка
            Frame31.StringGrid4.Cells[Shabl+3,  t6]:=FloatToStr(Trunc((Data[2]-
STabl3[8,t6])*100/Data[2]));  //площальостатка
            //рисование
            Doc2.ksText(0,Trunc(Data[5]+20),0,5,0,0,'Раскрой №'+FloatToStr(t6));
            //заполнениетаблицы
            for i2:=1 to Shabl do Doc2.ksText(0,(Data[5]+3+(i2-1)*3),0,2,0,0,'Шаблон №
'+FloatToStr(i2)+'; Кол-во: '+FloatToStr(STabl3[i2+2,t6]));
            Doc2.ksText(0,(Data[5]+3+(Shabl)*3),0,2,0,0,'Сост.  =
'+FloatToStr(Trunc(STabl3[8,t6]))+'  (мм2);  Заполнения  =  '+FloatToStr(Trunc((Data[2]-
STabl3[8,t6])*100/Data[2]))+' % ');
            Frame31.Image4.Canvas.Brush.Style:=bsSolid;
            Frame31.Image4.Canvas.Brush.Color:=clWhite;//$00F5F7F9;
            Frame31.Image4.Canvas.rectangle(0,0,width,height);
            Frame31.Image4.Width:=500;//325;//пох
            Frame31.Image4.Height:=500;//425;//по y
            Frame31.Image4.Canvas.pen.Color:=clBlack;//$00F5F7F9;
            Frame31.Image4.Canvas.MoveTo(0,0);    //по x
            Frame31.Image4.Canvas.LineTo(Width,0);
            Frame31.Image4.Canvas.MoveTo(0,0);  //по y
            Frame31.Image4.Canvas.LineTo(0,Height);
            if Data[4]>Data[5] then m:=Trunc(500/Data[4]) else m:=Trunc(500/Data[5]);
            u:=0;
            For u2:=6 to (Trunc(Data[0])*2+5) do//переборкоординат
        if u2 mod 2=0 then
             begin
               xu:=Trunc(Data[u2])*m+2;//том на 100 убрать тут чтоб видно было тк размер
примера мал
        yu:=Trunc(Data[u2+1])*m+2;
               PointArray[u]:=Point(xu,yu); u:=u+1;
               Frame31.Image4.Canvas.MoveTo(xu,0);  //по y
               Frame31.Image4.Canvas.LineTo(xu,5);
```

```
              Frame31.Image4.Canvas.MoveTo(0,yu);  //по y
              Frame31.Image4.Canvas.LineTo(5,yu);
              //рисованиевкомпас
              if      u2<(Trunc(Data[0])*2+5-1)   then   Doc2.ksLineSeg(      Trunc(Data[u2]),
Trunc(Data[u2+1]), Trunc(Data[u2+2]),  Trunc(Data[u2+3]), 1 );
              if      u2=(Trunc(Data[0])*2+5-1)   then   Doc2.ksLineSeg(      Trunc(Data[u2]),
Trunc(Data[u2+1]), Trunc(Data[6]),  Trunc(Data[7]), 1 );
            end;
           Frame31.Image4.Canvas.pen.Color:=$00E4E4E4;//clBlack;//clSilver;//фонДО
           Frame31.Image4.Canvas.Brush.Color:=$00E4E4E4;//clBlack;//clSilver;//фонДО
           Frame31.Image4.Canvas.Polygon(Slice(PointArray,u));//прорисовкаполигона
           u:=0;
           //рисованиешаблонов
           For u2:=(rs) to (Trunc(Raskroi3[rs][1])+rs-1) do
           begin
            t:=Trunc(Raskroi3[u2][2]);//номершаблона
            a1:=Trunc(Raskroi3[u2][5]);//угол
            for u3:=1 to Trunc(F[t,0]*2) do
             if u3 mod 2<>0 then//еслих
        begin
        //// по математическим правилам до ближайшего целого:
        xu:=Round(((F[t,u3]*cos(a1*3.14/180)-
F[t,u3+1]*sin(a1*3.14/180))+Raskroi3[u2][3]))*m+2;//x

yu:=Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi3[u2][4]))*m+2;//
y
              Frame31.Image4.Canvas.MoveTo(xu,0);  //по y
              Frame31.Image4.Canvas.LineTo(xu,5);
              Frame31.Image4.Canvas.MoveTo(0,yu);  //по y
              Frame31.Image4.Canvas.LineTo(5,yu);
              //ShowMessage(FloatToStr(u2)+'                                    координаты:
'+FloatToStr(xu)+';'+FloatToStr(yu));
              PointArray[u]:=Point(xu,yu); u:=u+1;
              //рисованиевкомпас
              if                u3<Trunc(F[t,0]*2-1)          then          Doc2.ksLineSeg(
Round(((F[t,u3]*cos(a1*3.14/180)-F[t,u3+1]*sin(a1*3.14/180))+Raskroi3[u2][3])),
Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi3[u2][4])),Round(((F[t,
u3+2]*cos(a1*3.14/180)-F[t,u3+3]*sin(a1*3.14/180))+Raskroi3[u2][3])),
Round(((F[t,u3+3]*cos(a1*3.14/180)+F[t,u3+2]*sin(a1*3.14/180))+Raskroi3[u2][4])), 1 );
              if                u3=Trunc(F[t,0]*2-1)          then          Doc2.ksLineSeg(
Round(((F[t,u3]*cos(a1*3.14/180)-F[t,u3+1]*sin(a1*3.14/180))+Raskroi3[u2][3])),
Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi3[u2][4])),Round(((F[t,
1]*cos(a1*3.14/180)-F[t,2]*sin(a1*3.14/180))+Raskroi3[u2][3])),
Round(((F[t,2]*cos(a1*3.14/180)+F[t,1]*sin(a1*3.14/180))+Raskroi3[u2][4])), 1 );
            end;
           Frame31.Image4.Canvas.pen.Color:=clBlack;//clSilver;//фонДО$00E4E4E4;//
           Frame31.Image4.Canvas.Brush.Color:=$00E4E4E4; // Россия
           Frame31.Image4.Canvas.pen.Color:=clBlack;//clSilver;//фонДО$00E4E4E4;//
           case Trunc(Raskroi3[u2][2]) of
            1:Frame31.Image4.Canvas.Brush.Color:=clGreen; // Россия
            2:Frame31.Image4.Canvas.Brush.Color:=clYellow; // Англия
            3:Frame31.Image4.Canvas.Brush.Color:=clRed; // Австрия
```

```
            4:Frame31.Image4.Canvas.Brush.Color:=clBlue; // Германия, Дания, Исландия, Нидерланды
             5:Frame31.Image4.Canvas.Brush.Color:=clBlack; // Италия
            end;
            Frame31.Image4.Canvas.Polygon(Slice(PointArray,u));//прорисовкаполигона
            u:=0;
          end;
          Frame31.Image4.Picture.SaveToFile(ProgDIR +'\show projects'+'\'+Frame21.edit1.text+'\'+'Nesting №3'+'\'+IntToStr(t6)+'.bmp');
        //добавление картинки из папки в таблицу для отображения как в галлереи
        Frame31.ShellListView5.Root:=ProgDIR +'\show projects'+'\'+Frame21.edit1.text+'\'+'Nesting №3';
        //Сохраняем в виде растрового изображения
          //Подготавливаем параметры сохранения в растр
        //RasterFormatParam  pPar;
          //RasterFormatParam RasterFormatParam;
          //RasterFormatParam.colorBPP := (BPP_COLOR_24);
          //RasterFormatParam.colorType := BLACKWHITE;
          { RasterFormatParam.extResolution := 300;
           RasterFormatParam.extScale := 1;
           RasterFormatParam.greyScale := true;
           RasterFormatParam.multiPageOutput := False;
           RasterFormatParam.onlyThinLine := false;
           RasterFormatParam.rangeIndex := 0;
           RasterFormatParam.format := FORMAT_BMP;
           Result := '.JPG';
           doc2.SaveAsToRasterFormat(('C:\Users\Сергей\Desktop\KR_Demo2.1\show projects'+'\'+Frame21.edit1.text+'\'+'Nesting №3'+'\'+IntToStr(t6)+Result), RasterFormatParam);
            }
          doc2.ksSaveDocument(ProgDIR +'\show projects'+'\'+Frame21.edit1.text+'\'+'Nesting №3'+'\'+IntToStr(t6)+'.cdw');
          //закрытиефрагмента
          if doc2<>nil then doc2:=nil;
          //ShowMessage('Раскрой = '+FloatToStr(t6));
          //ShowMessage('шаблоны i1='+FloatToStr(STabl2[3,t6])+' i2='+FloatToStr(STabl2[4,t6])+' i3='+FloatToStr(STabl2[5,t6])+' i4='+FloatToStr(STabl2[6,t6])+' i5='+FloatToStr(STabl2[7,t6]));
        m:=0;
          for j:=1 to 5 do m:=m-Ksh[4,j];//К для данного ДО
          k:=0;
          t6:=t6+1;
          //матрица ДО первоначального вида
        For j:=0 to (Trunc(Data[5])) do
            For j1:=0 to (Trunc(Data[4])) do
              Matr0[j][j1]:=Matr[j][j1];
         until  ((m=0));
         Frame101.ProgressBar1.Position:=1000;
         Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
         Frame101.Label5.Refresh;
         Frame101.ProgressBar1.Visible:=false;
         Frame31.StringGrid4.Cells[Shabl+2, t6]:='Итого:';
         Frame31.StringGrid4.Cells[Shabl+3, t6]:=FloatToStr(t6-1);  //всеговариантовраскроя
```

```
          if t6<25 then Frame31.StringGrid4.Height:=(3+t6)*20+5
          else Frame31.StringGrid4.Height:=(1+25)*20;
          Panel1.Enabled:=true;//заблокированныпанелькнопок
          SpeedButton4.Enabled:=false;
          Frame31.Visible:=true;
        //видимость фрама отображения анамации
          //Frame21.Visible:=false;
        Frame101.Visible:=false;
          //TabSheet.Visible:=true;
          Frame101.Label3.Caption:='';
          Frame101.Label3.Refresh;
          Frame101.Label4.Caption:='';
          Frame101.Label4.Refresh;
          if (t6-1)>=1 then
          begin
            Frame31.Image9.Picture.LoadFromFile(ProgDIR                    +'\show
projects\'+Frame21.edit1.text+'\'+'Nesting №3'+'\'+IntToStr(1)+'.bmp');
            Frame31.Image9.Proportional:=true;
            Frame31.Image9.Transparent:=true;
        end;
          h03:=1;
          h03max:=t6-1;
          //Закрытие компас в общем
        if Kompas <> nil then
          begin
            //принудительнозакрыть
            Kompas.Quit;
        Kompas := nil;
          end;
        ShowMessage('Для раскроя необходимого количества шаблонов'+#13+'Необходимо
'+FloatToStr(t6-1)+' лист(а/ов) делового остатка');
        end;
```

ПРИЛОЖЕНИЕ Н

Листинг реализации карт раскроя методом №2

```
procedure TForm1.SpeedButton5Click(Sender: TObject);
var
  i1,i2,i3,i4,i5,n:integer;
  Variant : array of array of integer;//вариантыпереборашаблонов

k0,t6,j45,x1,y2,x2,j1,j,t1,k1,k2,g,g1,g2A,g2B,g3A,g3B,g2,g3,dl,sh,kol,g33,k,j2,imax,t,j3,a1,yg,j
4,m:integer;
  Xmin,Xmax,Ymin,Ymax,t29,t30:real;
  c,h,x,S:real;
  Matr0 : array of array of integer;//матрицаДО
  a,b:double;
  max:double;
  u,u1,u2,xu,yu,u3,u4,u5,rasp,Raskr3,k9,k10:integer;
  V,F11,razrehenie:Boolean;
  Z: array [0..100,0..100] of real;  //массивскоординатамифигуры
  F1: array [1..15] of real;
  p:Integer;
begin
  p:=Form1.Width-200;
  Frame101.ProgressBar1.Width:=p;
  Panel1.Enabled:=false;//заблокированныпанелькнопок
  Frame31.PageControl1.ActivePageIndex:=4;
  Frame31.StringGrid5.ColCount:=4+Shabl;
  if Shabl<=2 then Frame31.StringGrid5.Width:=(4+Shabl)*65+10
  else  Frame31.StringGrid5.Width:=(4+2)*65+10;
  Frame31.Visible:=false;
//видимость фрама отображения анамации
  Frame31.StaticText5.Visible:=false;
Frame31.ShellListView3.Visible:=true;
  Frame101.Visible:=true;
  Frame101.ProgressBar1.Visible:=true;
  Frame101.ProgressBar1.Position:=0;
  Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
  Frame101.Label5.Refresh;
  Frame101.Refresh;
  Frame101.Label3.Caption:=''; Frame101.Label3.Refresh;
  Frame101.Label3.Caption:='Созданиеобъектаавтоматизации KOMPAS_Graphic';
  Frame101.Label3.Refresh;
  Frame101.Label4.Caption:=' ';
Frame101.Label4.Refresh;
  //рисование   в КОМПАС
  // Создать объект автоматизации KOMPAS_Graphic
  if Kompas = nil then
begin
    {$IFDEF __LIGHT_VERSION__}
    Kompas:= KompasObject( CreateOleObject('KompasLT.Application.5') );
    {$ELSE}
    Kompas:= KompasObject( CreateOleObject('Kompas.Application.5') );
    {$ENDIF}
```

```
  if Kompas <> nil then Kompas.Visible := false;    //невиднопокомпастконаненужна
 end;

 n:=0;
 case Shabl of
  5:
begin
   //освобождение памяти матрицы
   SetLength(Variant,(120+1),(Shabl+1));//выделение области памяти для матрицы
(у или строка макс, х или столбец максимально)
For i1:=1 to 5 do
    For i2:=1 to 5 do
     For i3:=1 to 5 do
      For i4:=1 to 5 do
       For i5:=1 to 5 do
        if
((i1<>i2)and(i1<>i3)and(i1<>i4)and(i1<>i5)and(i2<>i3)and(i2<>i4)and(i2<>i5)and(i3<>i4)and
(i3<>i5)and(i4<>i5))
          then
          begin
           n:=n+1;
           Variant[n,1]:=i1;
           Variant[n,2]:=i2;
           Variant[n,3]:=i3;
           Variant[n,4]:=i4;
           Variant[n,5]:=i5;
end;
   end;
   4:
   begin
    //освобождение памяти матрицы
    SetLength(Variant,(24+1),(Shabl+1));//выделение области памяти для матрицы
(у или строка макс, х или столбец максимально)
For i1:=1 to 4 do
    For i2:=1 to 4 do
     For i3:=1 to 4 do
      For i4:=1 to 4 do
       if ((i1<>i2)and(i1<>i3)and(i1<>i4)and(i2<>i3)and(i2<>i4)and(i3<>i4))
       then
       begin
        n:=n+1;
        Variant[n,1]:=i1;
        Variant[n,2]:=i2;
        Variant[n,3]:=i3;
        Variant[n,4]:=i4;
       end;
   end;
3:
   begin
    //освобождение памяти матрицы
    SetLength(Variant,(6+1),(Shabl+1));//выделение области памяти для матрицы
(у или строка макс, х или столбец максимально)
```

```
        For i1:=1 to 3 do
              For i2:=1 to 3 do
                For i3:=1 to 3 do
                  if ((i1<>i2)and(i1<>i3)and(i2<>i3))
                  then
                  begin
                    n:=n+1;
                    Variant[n,1]:=i1;
                    Variant[n,2]:=i2;
                    Variant[n,3]:=i3;
                    //ShowMessage('1 = '+FloatToStr(i1)+' 2 = '+FloatToStr(i2)+' 3 =
'+FloatToStr(i3));
        end;
           end;
           2:
           begin
             //освобождение памяти матрицы
             SetLength(Variant,(2+1),(Shabl+1));//выделение области памяти для матрицы
(у или строка макс, х или столбец максимально)
        n:=2;
             Variant[1,1]:=1;
             Variant[1,2]:=2;
             Variant[2,1]:=2;
             Variant[2,2]:=1;
        end;
           1:
           begin
             //освобождение памяти матрицы
             //SetLength(Variant,(1),(Shabl));//выделение области памяти для матрицы (у
или строка макс, х или столбец максимально)
        n:=1;
             j:=1;
             //Variant[1,1]:=1;
           end;
         end;
        //освобождение матрицы
          SetLength(STabl4,(100),(n+10));//выделение области памяти для матрицы (у или
строка макс, х или столбец максимально)
          SetLength(Raskroi4,(1500),(100));
          SetLength(Matr0,(Trunc(Data[5])+1000),(Trunc(Data[4])+1000));//выделение области
памяти для матрицы
        ForceDirectories(ProgDIR +'\show projects'+'\'+Frame21.edit1.text+'\'+'Nesting
№4');//созданиепапкисименем
          m:=0;
          t6:=0;
          t29:=1000/(n*Shabl*Data[5]*(Data[4]));
        t30:=0;
          //1. перебор от 1 до n вариантов раскроя
        For i1:=1 to n do
          begin
            Frame101.Label3.Caption:='Вариантраскроя №'+FloatToStr(i1);
        Frame101.Label3.Refresh;
```

```
          //онуление матрицы первоначального вида ДО//матрица ДО первоначального
вида
        For j:=0 to (Trunc(Data[5])) do
            For j1:=0 to (Trunc(Data[4])) do
              Matr0[j][j1]:=Matr[j][j1];
        //2. перебор вариантов шаблона в раскрое от 1 до Shabl
        for i:=1 to (Shabl) do
           begin
            if Shabl>1 then j:=Variant[i1,i]//номерраскрварианташаблона
            else j:=1;
            Frame101.Label4.Caption:='              Вмещениешаблона              '+FloatToStr(j)+'
видапомаксимуму';
            Frame101.Label4.Refresh;
            //вмещениешаблона
            k0:=0;
        //перебор матрицы для вставки шаблона
            For k1:=(0) to Trunc(Data[5]) do//Y
                        begin
             //Frame101.ProgressBar1.Refresh;
                          for k2:=(0) to Trunc(Data[4]) do //X
            begin
                Frame101.ProgressBar1.Position:=Trunc(t29+t30);
                Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
                Frame101.Label5.Refresh;
                t30:=t29+t30;
        //проверка войдет ли в до
                //обнуление матриц  Z[g,0...100]  for g:=0 to 100 do
        For t:=0 to 50 do
                  For t1:=0 to 50 do
                    Z[t,t1]:=0;
                if Matr0[k1][k2]=1 then
                begin
                  a1:=360;
                  f11:=False;
                  repeat
                    razrehenie:=true;
                    if ((F11=true))
                    then
                    begin
                      g1:=1;
                      //отражениефигуры
                      for g:=Trunc(F[j,0]*2) downto 1 do
                        if g mod 2=0 then  F1[g1+1]:=F[j,g]         //если y
                        else
                        begin
                          F1[g1]:=F[j,g]-2*F[j,g]+DataF[j][3];
        g1:=g1+2;
                        end;  //если x
                      //сохранили коорлинаты нового положения фигуры
        for g:=1 to Trunc(F[j,0]*2) do
                        ifgmod 2<>0 then//еслих
        begin//по математическим правилам до ближайшего целого:
```

```
        Z[0,g]:=Round(F1[g]*cos(a1*3.14/180)-F1[g+1]*sin(a1*3.14/180));//x   //k2
                        Z[0,g+1]:=Round(F1[g+1]*cos(a1*3.14/180)+F1[g]*sin(a1*3.14/180));//y
//k1
        //вставить проверку точек перегиба, если в данных координата в матрице MATP0=0
то =фалсе
        k9:=Round(Z[0,g]+k2); //x
                        k10:=Round(Z[0,g+1]+k1); //y
                        if NOT ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and
(k9<=Data[4])))) then begin razrehenie:=false; break; end;
                        if ((((k9>0) and (k10>0))and((k10<Data[5]) and (k9<Data[4])))) then
                        if Matr0[k10][k9]=0 then begin razrehenie:=false; break; end;
        end;
                    end
                    else//сохранили коорлинаты нового положения фигуры без
отражения//сохранили коорлинаты нового положения фигуры
        for g:=1 to Trunc(F[j,0]*2) do
                      if g mod 2<>0 then//еслих
        begin
        //// по математическим правилам до ближайшего целого:
        Z[0,g]:=Round(F[j,g]*cos(a1*3.14/180)-F[j,g+1]*sin(a1*3.14/180));//x
                        Z[0,g+1]:=Round(F[j,g+1]*cos(a1*3.14/180)+F[j,g]*sin(a1*3.14/180));//y
                        k9:=Round(Z[0,g]+k2); //x
                        k10:=Round(Z[0,g+1]+k1); //y
                        if NOT ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and
(k9<=Data[4])))) then begin razrehenie:=false; break; end;
                        if ((((k9>0) and (k10>0))and((k10<Data[5]) and (k9<Data[4])))) then
                        if Matr0[k10][k9]=0 then begin razrehenie:=false; break; end;
                       end;
                    if razrehenie=true then
                    begin
        g1:=0;//прямых
                     //заполнение уравнений прямых в фигуре
        for g:=1 to Trunc(F[j,0]*2) do
                      if g mod 2<>0 then//еслих
                      begin
                        g1:=g1+1;
                        if g=F[j,0]*2-1 then
                        begin
                          Z[g1,1]:=Z[0,2]-Z[0,g+1];//a
                          Z[g1,2]:=Z[0,g]-Z[0,1] ;   //b
                          Z[g1,3]:=Z[0,g+1]*(Z[0,1]-Z[0,g])-Z[0,g]*(Z[0,2]-Z[0,g+1]) ;      //c
        if Z[0,1]<Z[0,3] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                         else  if Z[0,1]>Z[0,3] then Z[g1,0]:=-1; //знак (в какую сторону вектор)
                         if Z[0,1]=Z[0,3] then
                           if Z[0,1]<Z[0,5] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                           else   if Z[0,1]>Z[0,5] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
        end
                      else
                      begin
                        Z[g1,1]:=Z[0,g+3]-Z[0,g+1] ;//a
                        Z[g1,2]:=Z[0,g]-Z[0,g+2] ;   //b
```

```
                    Z[g1,3]:=Z[0,g+1]*(Z[0,g+2]-Z[0,g])-Z[0,g]*(Z[0,g+3]-Z[0,g+1])        ;
//c
      if Z[0,g]<Z[0,g+2] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                    else  if Z[0,g]>Z[0,g+2] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
                    if Z[0,g]=Z[0,g+2] then
                     if Z[0,g]<Z[0,g+4] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                     else  if Z[0,g]>Z[0,g+4] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
                    end;
                   end;
                  //поиск минимума максимума новых координат фигуры
                  Xmin:=Z[0][1];
                  Xmax:=Z[0][1];
                  Ymin:=Z[0][2];
      Ymax:=Z[0][2];
                  For g:=3 to Trunc(F[j,0]*2) do
                  begin
                   if g mod 2<>0 then
                   begin
      if Z[0][g]<=Xmin then Xmin:=Z[0][g];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
      if Z[0][g]>Xmax then  Xmax:=Z[0][g];
                   end
                   // Поиск Ymin,Ymax
                   else
                   begin
      if Z[0][g]<=Ymin then Ymin:=Z[0][g];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
      if Z[0][g]>Ymax then  Ymax:=Z[0][g];
                   end;
                  end;
                  //онулениематрицышаблона
                  For y2:=Trunc(Ymin) to Trunc(Ymax) do
                   For x2:=Trunc(Xmin) to Trunc(Xmax) do
      MatrSH[y2][x2]:=0;
                  //поиск матрицы фигуры по найденным координатам
                  //сохраняем матрицу шаблона в MatrSH[y2][x2]
      V:=false;
                  for y2:=Trunc(Ymin) to Trunc(Ymax) do
                  begin
                   g:=0;//сколько 1 встроке
                   for x2:=Trunc(Xmin) to Trunc(Xmax) do
      begin
                    V:=false;
                    //перебор прямых, посмотреть сколько пересечений,
                    for g2:=1 to Trunc(F[j,0]) do    //проверка лежит ли точка на осях
шаблона
      begin
                     j45:=Trunc(x2*Z[g2,1]+y2*Z[g2,2]+Z[g2,3]);
```

```
                        if                                                  (j45=0)//and
(((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1])))
//or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1]))
                        then
                        begin
                          if
((g2=Trunc(F[j,0]))and((y2>=Z[0,2])and(y2<=Z[0,g2*2])and(x2>=Z[0,1])and(x2<=Z[0,g2*2-
1])))
                          then
                          begin
                            MatrSH[y2][x2]:=1;
                            V:=true;
                            break;
                          end
                          else
                            if ((Z[g2,1]=0)and(Z[g2,3]=0))
        then   //если прямая параллельна оси X
        begin
                              if (Z[0,g2*2-1]>Z[0,g2*2+1])and(g2<>1)then
                                if ((x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1]))
                                then
                                begin
                                  MatrSH[y2][x2]:=1;
                                  V:=true;
                                  break;
                                end
                                else if (Z[0,g2*2-1]<Z[0,g2*2+1])and(g2<>1)then
                                  if ((x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1]))
                                  then
                                  begin
                                    MatrSH[y2][x2]:=1;
        V:=true;
                                    break;
                                  end;
                            end
                            else //если не паралельная и не конечная то
        if((Z[g2,1]<>0)and(Z[g2,3]<>0))and((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z
[0,g2*2-
1])and(x2<=Z[0,g2*2+1]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-
1])and(x2>=Z[0,g2*2+1]))
                              then
                              begin
                                MatrSH[y2][x2]:=1;
                                V:=true;
                                break;
                              end;
                          end;   //
                        end;
                        if (V=false) then
                        begin
        j1:=0;
```

```
                    SetLength(coord,Trunc(F[j,0])+100);//выделение памяти для
динамического массива
                    for g2:=1 to Trunc(F[j,0]) do  //проверка лежит ли точка на осях
шаблона
        begin
                     if
(((Z[g2,1]<>0)and(Z[g2,2]<>0))or((Z[g2,1]<>0)and(Z[g2,2]=0)))and(((y2>=Z[0,g2*2])and(y2<
=Z[0,g2*2+2]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2]))) then  //
                     begin
                      if     ((g2=Trunc(F[j,0]))and((Z[0,2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,1])<>0))  or  ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])<>0))or ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])=0))
                      then
                      begin
                       x1:=Trunc((Z[g2,2]*y2+Z[g2,3])/(-Z[g2,1]));
                       if x1<=x2 then
                       begin
                        if j1=0 then
        begin
                         j1:=j1+1;//сколько точек пересечения до прямой x=0
        coord[j1]:=x1;
                       end
                       else
                       begin
                        cr1:=true;
                        for cr:=1 to j1 do
                         if coord[cr]=x1 then begin cr1:=false; break; end;
        if cr1=true then
                        begin
                         j1:=j1+1;//сколько точек пересечения до прямой x=0
        coord[j1]:=x1;
                        end;
                       end;
                      end;
                     end;
                     if j1 mod 2<>0 then MatrSH[y2][x2]:=1  //еслинечетное
                     else MatrSH[y2][x2]:=0;  //есличетное
                    end;
        end;   //x
                end;    //y
                //наложение матрицы фигуры на матрицу ДО
        if
((Xmin+k2)>=0)and((Xmax+k2)<=Trunc(Data[4]))and((Ymin+k1)>=0)and((Ymax+k1)<=Trun
c(Data[5]))
                then
                begin
                 V:=true;
                 for g:=Trunc(Ymin) to Trunc(Ymax) do
                 begin
```

```
                        for g1:=Trunc(Xmin) to Trunc(Xmax) do
                          if MatrSH[g][g1]=1 then  if MatrSH[g][g1]<>Matr0[g+k1][g1+k2] then
begin V:=false;  break; end;     //
                          if V=false then break;
                        end;
                      end
                      else V:=false;
                      if V=true then
                      begin
                        //ShowMessage('онудениематрицыеслифигуравходит');
                                    for g:=Trunc(Ymin) to Trunc(Ymax) do
                          for g1:=Trunc(Xmin) to Trunc(Xmax) do
                            if MatrSH[g][g1]=1 then Matr0[g+k1][g1+k2]:=0;
        Matr0[k1][k2]:=1;
                        //сохранение координат
                        k:=k+1; //всего шаблонов в раскрое
                        k0:=k0+1; //шаблонов данного вида на данном шаблоне
        Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
                        Frame101.Label4.Caption:='Вмещениешаблона '+FloatToStr(j)+'
видапомаксимуму. Вместили '+FloatToStr(k0)+' шт.';
                        Frame101.Label4.Refresh;
                        //ShowMessage('вписали '+FloatToStr(k)+'; осталось
'+FloatToStr(Ksh[4,i]));
                        if Shabl>1 then Raskroi4[k+rs-
1,2]:=Variant[i1,i]//номерраскрварианташаблона
        else Raskroi4[k+rs-1,2]:=1;//сохранение номера шаблона
        Raskroi4[k+rs-1,3]:=k2;//x0
                        Raskroi4[k+rs-1,4]:=k1;//y0
        Raskroi4[k+rs-1,5]:=a1;//ориентация(угол поворота относительно x0y0 против
часовой стрелки
                        STabl4[1,t6+1]:=1;  //перебор № ДО
                        STabl4[2,t6+1]:=t6+1;  //перебор № листа
                        STabl4[j+2,t6+1]:=k0;
                        break;//прерывание поиска угла
                      end;
                    end;//КОНЕЦ действие если разрешение:=true;
        if  DataF[j][8]<>2 then
                      if
((Frame31.LabeledEdit1.Text<>'')and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                      else yg:=90;
                      if DataF[j][8]=2 then
                      begin
                        if (DataF[j][3]<>DataF[j][5]) then//еслипрямоугольник
                        begin
                          if Frame31.CheckBox1.Checked = false
then//есликлаишазапретаненажатато
                            if
((Frame31.LabeledEdit1.Text<>'')and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
```

```
                    else yg:=90;
                   if Frame31.CheckBox1.Checked = true then yg:=90;
                  end;
                  if (DataF[j][3]=DataF[j][5]) then//если rdf
                  begin
                   if Frame31.CheckBox1.Checked = false
then//есликлаишазапретаненажатато
                    if
((Frame31.LabeledEdit1.Text<>")and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                    else yg:=360;
                   if Frame31.CheckBox1.Checked = true then yg:=360;
                  end;
                 end;
                 a1:=a1-yg;
                until ((a1=0));
                if ((V=true)) then Matr0[k1][k2]:=0;
               end;//если =1
             end; //x
            end; //y
            //ShowMessage('вариант: '+FloatToStr(t6+1)+' шаблвида '
+FloatToStr(Variant[i1,i])+' = '+FloatToStr(k0)+' шт');
        end;//конец 2
          V:=true;
          //Проверка, были ли уже такие комбинации:
        if (t6>0) then
           for i2:=1 to t6 do
            if
((STabl4[3,t6+1]=STabl4[3,i2])and(STabl4[4,t6+1]=STabl4[4,i2])and(STabl4[5,t6+1]=STabl4[5
,i2])and(STabl4[6,t6+1]=STabl4[6,i2])and(STabl4[7,t6+1]=STabl4[7,i2])) then
              begin
               V:=false;
               for u2:=0 to 8 do  STabl4[u2,t6+1]:=0;
        k:=0;
               k0:=0;
               //матрица ДО первоначального вида
        For j:=0 to (Trunc(Data[5])) do
              For j1:=0 to (Trunc(Data[4])) do
               Matr0[j][j1]:=Matr[j][j1];
              t6:=t6;
              break;
             end;
          if (V=true)  then
          begin   {}
           //сохранение
           t6:=t6+1;
           Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
           Frame101.Label4.Caption:='Раскрой '+FloatToStr(t6)+'. Сохранение данных.';
           Frame101.Label4.Refresh;
           Raskroi4[rs,1]:=k;
```

```
            STabl4[8,t6]:=Data[2]-
(DataF[1][1]*STabl4[3,t6]+DataF[2][1]*STabl4[4,t6]+DataF[3][1]*STabl4[5,t6]+DataF[4][1]*S
Tabl4[6,t6]+DataF[5][1]*STabl4[7,t6]);//остаток от раскроя (по  ПРЯМОУГОЛЬНИКАМ)

        Frame31.StringGrid5.Cells[0, 0]:='№';
            Frame31.StringGrid5.Cells[1, 0]:='ДО';
            for i2:=1 to Shabl do Frame31.StringGrid5.Cells[i2+1,  0]:=IntToStr(i2);
//видшаблона, использовавшегоприраскрое;
            Frame31.StringGrid5.Cells[Shabl+2, 0]:='S ост.';  //площальостатка
            Frame31.StringGrid5.Cells[Shabl+3, 0]:='% заполнения';  //площальостатка
            //заполнениетаблицы
            Frame31.StringGrid5.Cells[0,
t6]:=FloatToStr(STabl4[2,t6]);//номервариантараскроя
            Frame31.StringGrid5.Cells[1, t6]:=FloatToStr(STabl4[1,t6]);  //видделовогоостатка,
использовавшегоприраскрое
            for i2:=1 to Shabl do Frame31.StringGrid5.Cells[i2+1,
t6]:=FloatToStr(STabl4[i2+2,t6]);  //видшаблона, использовавшегоприраскрое
//видшаблона, использовавшегоприраскрое;
            Frame31.StringGrid5.Cells[Shabl+2,  t6]:=FloatToStr(Trunc(STabl4[8,t6]));
//площальостатка
            Frame31.StringGrid5.Cells[Shabl+3,  t6]:=FloatToStr(Trunc((Data[2]-
STabl4[8,t6])*100/Data[2]));  //площальостатка
        //рисование  в КОМПАС
            // Создать объект автоматизации KOMPAS_Graphic  ранее
            //Получаем и заполняем интерфейс параметров документа
        DocumentParam:=ksDocumentParam(kompas.GetParamStruct(ko_DocumentParam));
            DocumentParam.Init();
            DocumentParam.type_:=ksDocumentFragment;//Drawing;
            //Получаеминтерфейсдокумента
            Doc2:=ksDocument2D(kompas.Document2D);
            //Создаемновыйчертеж
            Doc2.ksCreateDocument(DocumentParam);
            //рисованиев image
            Frame31.Image5.Canvas.Brush.Style:=bsSolid;
            Frame31.Image5.Canvas.Brush.Color:=clWhite;//$00F5F7F9;
            Frame31.Image5.Canvas.rectangle(0,0,width,height);
            Frame31.Image5.Width:=500;//325;//пох
            Frame31.Image5.Height:=500;//425;//по y
            Frame31.Image5.Canvas.pen.Color:=clBlack;//$00F5F7F9;
            Frame31.Image5.Canvas.MoveTo(0,0);    //по x
            Frame31.Image5.Canvas.LineTo(Width,0);
            Frame31.Image5.Canvas.MoveTo(0,0);  //по y
            Frame31.Image5.Canvas.LineTo(0,Height);
            if Data[4]>Data[5] then m:=Trunc(500/Data[4]) else m:=Trunc(500/Data[5]);
            u:=0;
            For u2:=6 to (Trunc(Data[0])*2+5) do//переборкоординат
        if u2 mod 2=0 then
             begin
               xu:=Trunc(Data[u2])*m+2;//том на 100 убрать тут чтоб видно было тк размер
примера мал
        yu:=Trunc(Data[u2+1])*m+2;
              PointArray[u]:=Point(xu,yu); u:=u+1;
```

```
                Frame31.Image5.Canvas.MoveTo(xu,0);  //по y
                Frame31.Image5.Canvas.LineTo(xu,5);
                Frame31.Image5.Canvas.MoveTo(0,yu);  //по y
                Frame31.Image5.Canvas.LineTo(5,yu);
                //рисованиевкомпас
                if      u2<(Trunc(Data[0])*2+5-1)  then  Doc2.ksLineSeg(      Trunc(Data[u2]),
Trunc(Data[u2+1]), Trunc(Data[u2+2]),  Trunc(Data[u2+3]), 1 );
                if      u2=(Trunc(Data[0])*2+5-1)  then  Doc2.ksLineSeg(      Trunc(Data[u2]),
Trunc(Data[u2+1]), Trunc(Data[6]),  Trunc(Data[7]), 1 );
              end;
              Frame31.Image5.Canvas.pen.Color:=$00E4E4E4;//clBlack;//clSilver;//фонДО
              Frame31.Image5.Canvas.Brush.Color:=$00E4E4E4;//clBlack;//clSilver;//фонДО
              Frame31.Image5.Canvas.Polygon(Slice(PointArray,u));//прорисовкаполигона
              u:=0;
              //рисованиешаблонов
              For u2:=(rs) to (Trunc(Raskroi4[rs][1])+rs-1) do
              begin
                t:=Trunc(Raskroi4[u2][2]);//номершаблона
                a1:=Trunc(Raskroi4[u2][5]);//угол
                for u3:=1 to Trunc(F[t,0]*2) do
                  if u3 mod 2<>0 then//еслих
        begin
        //// по математическим правилам до ближайшего целого:
        xu:=Round(((F[t,u3]*cos(a1*3.14/180)-
F[t,u3+1]*sin(a1*3.14/180))+Raskroi4[u2][3]))*m+2;//x //начальныеточки

yu:=Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi4[u2][4]))*m+2;//
y
                  Frame31.Image5.Canvas.MoveTo(xu,0);  //по y
                  Frame31.Image5.Canvas.LineTo(xu,5);
                  Frame31.Image5.Canvas.MoveTo(0,yu);  //по y
                  Frame31.Image5.Canvas.LineTo(5,yu);
                  //ShowMessage(FloatToStr(u2)+'                                    координаты:
'+FloatToStr(xu)+';'+FloatToStr(yu));
                  PointArray[u]:=Point(xu,yu); u:=u+1;
                  //рисованиевкомпас
                  if                u3<Trunc(F[t,0]*2-1)        then        Doc2.ksLineSeg(
Round(((F[t,u3]*cos(a1*3.14/180)-F[t,u3+1]*sin(a1*3.14/180))+Raskroi4[u2][3])),
Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi4[u2][4])),Round(((F[t,
u3+2]*cos(a1*3.14/180)-F[t,u3+3]*sin(a1*3.14/180))+Raskroi4[u2][3])),
Round(((F[t,u3+3]*cos(a1*3.14/180)+F[t,u3+2]*sin(a1*3.14/180))+Raskroi4[u2][4])), 1 );
                  if                u3=Trunc(F[t,0]*2-1)        then        Doc2.ksLineSeg(
Round(((F[t,u3]*cos(a1*3.14/180)-F[t,u3+1]*sin(a1*3.14/180))+Raskroi4[u2][3])),
Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi4[u2][4])),Round(((F[t,
1]*cos(a1*3.14/180)-F[t,2]*sin(a1*3.14/180))+Raskroi4[u2][3])),
Round(((F[t,2]*cos(a1*3.14/180)+F[t,1]*sin(a1*3.14/180))+Raskroi4[u2][4])), 1 );
                end;
              xu:=Round(((F[t,1]*cos(a1*3.14/180)-
F[t,1+1]*sin(a1*3.14/180))+Raskroi4[u2][3]));//x //начальныеточки

yu:=Round(((F[t,1+1]*cos(a1*3.14/180)+F[t,1]*sin(a1*3.14/180))+Raskroi4[u2][4]));//y
              Frame31.Image5.Canvas.pen.Color:=clBlack;//clSilver;//фонДО$00E4E4E4;//
```

```
                Frame31.Image5.Canvas.Brush.Color:=$00E4E4E4; // Россия
                Frame31.Image5.Canvas.pen.Color:=clBlack;//clSilver;//фонДО$00E4E4E4;//
                case Trunc(Raskroi4[u2][2]) of
                  1:Frame31.Image5.Canvas.Brush.Color:=clGreen; // Россия
                  2:Frame31.Image5.Canvas.Brush.Color:=clYellow; // Англия
                  3:Frame31.Image5.Canvas.Brush.Color:=clRed; // Австрия
                  4:Frame31.Image5.Canvas.Brush.Color:=clBlue; // Германия, Дания, Исландия,
Нидерланды
                  5:Frame31.Image5.Canvas.Brush.Color:=clBlack; // Италия
                end;
                Frame31.Image5.Canvas.Polygon(Slice(PointArray,u));//прорисовкаполигона
                u:=0;
              end;
            Doc2.ksText(0,Trunc(Data[5]+20),0,5,0,0,'Раскрой №'+FloatToStr(t6));
            //заполнениетаблицы
            for i2:=1 to Shabl do
            Doc2.ksText(0,(Data[5]+3+(i2-1)*3),0,2,0,0,'Шаблон № '+FloatToStr(i2)+'; Кол-во:
'+FloatToStr(STabl4[i2+2,t6]));
            Doc2.ksText(0,(Data[5]+3+(Shabl)*3),0,2,0,0,'Sост.                                =
'+FloatToStr(Trunc(STabl4[8,t6]))+' (мм2); Заполнения = '+FloatToStr(Trunc((Data[2]-
STabl4[8,t6])*100/Data[2]))+' % ');
            Frame31.Image5.Picture.SaveToFile(ProgDIR                                +'\show
projects'+'\'+Frame21.edit1.text+'\'+'Nesting №4'+'\'+IntToStr(t6)+'.bmp');
            //закрытиекомпас
            //сохранение
            doc2.ksSaveDocument(ProgDIR +'\show projects'+'\'+Frame21.edit1.text+'\'+'Nesting
№4'+'\'+IntToStr(t6)+'.cdw');
            //                  doc2.ksSaveDocument('C:\Users\Сергей\Desktop\KR_Demo2.1\show
projects'+'\'+Frame21.edit1.text+'\'+'Nesting №4'+'\'+IntToStr(t6)+'.bmp');
            //закрытиефрагмента
            if doc2<>nil then doc2:=nil;
        k:=0;
            //матрица ДО первоначального вида
            For j:=0 to (Trunc(Data[5])) do
        For j1:=0 to (Trunc(Data[4])) do
              Matr0[j][j1]:=Matr[j][j1];
        end;
           {else //матрица ДО первоначального вида
        For j:=0 to (Trunc(Data[5])) do
             For j1:=0 to (Trunc(Data[4])) do
              Matr0[j][j1]:=Matr[j][j1];   }
          end;//конец 1
          Frame31.StringGrid5.Cells[Shabl+2, t6+1]:='Итого:';
          Frame31.StringGrid5.Cells[Shabl+3, t6+1]:=FloatToStr(t6);  //всеговариантовраскроя
          if t6<25 then Frame31.StringGrid5.Height:=(3+t6)*20+5
          else Frame31.StringGrid5.Height:=(1+25)*20;
          Frame101.ProgressBar1.Position:=1000;
          Frame101.ProgressBar1.Refresh;
          Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
          Frame101.Label5.Refresh;
          Frame31.ShellListView3.Root:=ProgDIR                                +'\show
projects'+'\'+Frame21.edit1.text+'\'+'Nesting №4';
```

```
          if (t6)>=1 then
          begin
            Frame31.Image10.Picture.LoadFromFile(ProgDIR                    +'\show
projects\'+Frame21.edit1.text+'\'+'Nesting №4'+'\'+IntToStr(1)+'.bmp');
            Frame31.Image10.Proportional:=true;
            Frame31.Image10.Transparent:=true;
          end;
          h01:=1;
          h01max:=t6;
          Panel1.Enabled:=true;//заблокированныпанелькнопок
          SpeedButton5.Enabled:=false;
          Frame31.Visible:=true;
          //видимостьфрамаотображенияанамации
          Frame101.ProgressBar1.Visible:=False;
          Frame101.Visible:=false;
          Frame101.Label3.Caption:=' ';
          Frame101.Label3.Refresh;
          Frame101.Label4.Caption:=' ';
        Frame101.Label4.Refresh;
          //Закрытие компас в общем
        if Kompas <> nil
          then
          begin
            //принудительнозакрыть
            Kompas.Quit;
        Kompas := nil;
          end;
        end;
```

ПРИЛОЖЕНИЕ О

Листинг реализации карт раскроя методом №3

```
procedure TForm1.SpeedButton3Click(Sender: TObject);
var

k9,k10,j45,x1,y2,x2,j1,j,t1,k1,k2,g,g1,g2A,g2B,g3A,g3B,g2,g3,dl,sh,kol,g33,k,j2,i1,i2,i3,i4,i5,i
max,t,j3,a1,yg,j4,m, e1, e2, e3, e4, e5:integer;
  Xmin,Xmax,Ymin,Ymax,t29,t30:real;
  c,h,x,S:real;
  Matr0 : array of array of integer;//матрицаДО
  a,b:double;
  max:double;
  Ksh: array [1..100] of integer;
  L: array [1..7] of integer;
  u,u1,u2,xu,yu,u3,u4,u5,rasp,Raskr3,i0:integer;
  V,F11,razrehenie:Boolean;
  Z: array [0..100,0..100] of real;  //массивскоординатамифигуры
  F1: array [1..15] of real;
  p:Integer;
begin
  p:=Form1.Width-200;
  Frame101.ProgressBar1.Width:=p;
  Panel1.Enabled:=false;//заблокированныпанелькнопок
  Frame31.Visible:=false;
  Frame31.PageControl1.ActivePageIndex:=2;
  Frame31.StringGrid2.ColCount:=4+Shabl;
  if Shabl<=2 then Frame31.StringGrid2.Width:=(4+Shabl)*65+10
  else  Frame31.StringGrid2.Width:=(4+2)*65+10;
  Frame31.StaticText3.Visible:=false;
  Frame31.ShellListView4.Visible:=true;
  Frame101.Visible:=true;
  Frame101.ProgressBar1.Visible:=true;
  Frame101.ProgressBar1.Position:=0;
  Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
  Frame101.Label5.Refresh;
  Frame101.Refresh;
  Frame101.Label3.Caption:=''; Frame101.Label3.Refresh;
  Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
  Frame101.Label3.Caption:='Созданиеобъектаавтоматизации KOMPAS_Graphic';
  Frame101.Label3.Refresh;
  Frame101.Label4.Caption:=' ';
Frame101.Label4.Refresh;
  //рисование   в КОМПАС
  // Создать объект автоматизации KOMPAS_Graphic
  if Kompas = nil then
begin
    {$IFDEF __LIGHT_VERSION__}
    Kompas:= KompasObject( CreateOleObject('KompasLT.Application.5') );
    {$ELSE}
    Kompas:= KompasObject( CreateOleObject('Kompas.Application.5') );
    {$ENDIF}
```

```
          if Kompas <> nil then
            Kompas.Visible := false;    //невиднопокомпастконаненужна
          end;
          //ShowMessage('Начатьраскрой?');
          //создаетсяпапкасраскроями
          ForceDirectories(ProgDIR +'\show projects'+'\'+Frame21.edit1.text+'\'+'Nesting №2');//созданиепапкисименем
        for j:=1 to 15 do Ksh[j]:=0; //обнуление К для данного ДО
          //Этап 1: Поиск "К" (сколько шаблонов вида j войдет на ДО  //перебор вариантов раскроя
          SetLength(Matr0,(Trunc(Data[5])+1000),(Trunc(Data[4])+1000));//выделение области памяти для матрицы
          sec:=0;
          //ShowMessage('Поиск "К" '); расчет по максимальному кол-ву шаблонов на ДО
        t29:=150/(Shabl*Data[5]);
          t30:=0;
          Frame101.Label3.Caption:='  '; Frame101.Label3.Refresh;
        Frame101.Label3.Caption:='Расчет максимальго кол-ва шаблонов, входящих на деловой остаток';
        Frame101.Label3.Refresh;
          for i:=1 to (Shabl) do
          begin
            k:=0;
            Frame101.Label4.Caption:=' '; Frame101.Label4.Refresh;
            Frame101.Label4.Caption:='Вмещениешаблона '+FloatToStr(i)+' вида.';
            Frame101.Label4.Refresh;
              For j:=0 to (Trunc(Data[5])) do
                For j1:=0 to (Trunc(Data[4])) do
                  Matr0[j][j1]:=Matr[j][j1];
            //ShowMessage(FloatToStr(a)); //сколько
            //ShowMessage(FloatToStr(b));
            For k1:=(0) to Trunc(Data[5]) do//Y   //переборданныхДО
                        BEGIN
              Frame101.ProgressBar1.Position:=Trunc(t29+t30);
              t30:=t29+t30;
              Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
              Frame101.Label5.Refresh;
                          for k2:=(0) to Trunc(Data[4]) do //X
        begin
              if Matr0[k1,k2]=1 then          //надо НЕ РАВНО 0 те РАВНО 1
                                  begin
                if Matr0[k1][k2]=1 then
                begin
                  a1:=360;//угол
                  yg:=90;
                  f11:=False;    ////////////////////////////////////
                  repeat
                    razrehenie:=true;
                    //ShowMessage('Угол:                  '+FloatToStr(a1)+'                  источки ('+FloatToStr(k2)+','+FloatToStr(k1)+')');
                    if F11=true then
                    begin
```

```
                    g1:=1;
                    //отражениефигуры
                    for g:=Trunc(F[i,0]*2) downto 1 do
                    if g mod 2=0 then  F1[g1+1]:=F[i,g]         //если y
                    else
                    begin
                      F1[g1]:=F[i,g]-2*F[i,g]+DataF[i][3];
                      //ShowMessage('Точка'+FloatToStr(g1)+':
'+FloatToStr(F1[g1])+'Точка'+FloatToStr(g1+1)+': '+FloatToStr(F1[g1+1]));
        g1:=g1+2;
                    end;  //если x
                    //сохранили коорлинаты нового положения фигуры
        for g:=1 to Trunc(F[i,0]*2) do
                      ifgmod 2<>0 then//еслих
        begin//по математическим правилам до ближайшего целого:
        Z[0,g]:=Round(F1[g]*cos(a1*3.14/180)-F1[g+1]*sin(a1*3.14/180));//x
                      Z[0,g+1]:=Round(F1[g+1]*cos(a1*3.14/180)+F1[g]*sin(a1*3.14/180));//y
                      k9:=Round(Z[0,g]+k2); //x
                      k10:=Round(Z[0,g+1]+k1); //y
                      if NOT ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and (k9<=Data[4]))))
then begin razrehenie:=false; break; end;
                      if ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and (k9<=Data[4])))) then
                        if Matr0[k10][k9]=0 then begin razrehenie:=false; break; end;
                      //ShowMessage('1.                                    координаты:
'+FloatToStr(Z[0,g])+';'+FloatToStr(Z[0,g+1]));
        end;
                  end
                  else//сохранили коорлинаты нового положения фигуры без отражения
                  begin
                    //ShowMessage('поиск координатов нового положения фигуры без
отражения');
        for g:=1 to Trunc(F[i,0]*2) do
                    ifgmod 2<>0 then//еслих
        begin//по математическим правилам до ближайшего целого:
        Z[0,g]:=Round(F[i,g]*cos(a1*3.14/180)-F[i,g+1]*sin(a1*3.14/180));//x
                      Z[0,g+1]:=Round(F[i,g+1]*cos(a1*3.14/180)+F[i,g]*sin(a1*3.14/180));//y
                      k9:=Round(Z[0,g]+k2); //x
                      k10:=Round(Z[0,g+1]+k1); //y
                      if NOT ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and (k9<=Data[4]))))
then begin razrehenie:=false; break; end;
                      if ((((k9>0) and (k10>0))and((k10<Data[5]) and (k9<Data[4])))) then
                        if Matr0[k10][k9]=0 then begin razrehenie:=false; break; end;
                        //ShowMessage('угол'+FloatToStr(a1)+'                    координаты:
'+FloatToStr(Z[0,g])+';'+FloatToStr(Z[0,g+1]));
                    end;
                  end;
                  if razrehenie=true then
        begin
                    g1:=0;//прямых
                    //заполнение уравнений прямых в фигуре
                    //ShowMessage('заполнение уравнений прямых в фигуре');
        for g:=1 to Trunc(F[i,0]*2) do
```

```
                    if g mod 2<>0 then//еслих
                    begin
                      g1:=g1+1;
                      if g=F[i,0]*2-1 then
                      begin
                        Z[g1,1]:=Z[0,2]-Z[0,g+1] ;//a
                        Z[g1,2]:=Z[0,g]-Z[0,1] ;   //b
                        Z[g1,3]:=Z[0,g+1]*(Z[0,1]-Z[0,g])-Z[0,g]*(Z[0,2]-Z[0,g+1]) ;      //c
        if Z[0,1]<Z[0,3] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                        else  if Z[0,1]>Z[0,3] then Z[g1,0]:=-1; //знак (в какую сторону вектор)
                        if Z[0,1]=Z[0,3] then
                          if Z[0,1]<Z[0,5] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                          else  if Z[0,1]>Z[0,5] then Z[g1,0]:=-1; //знак (в какую сторону вектор)
        end
                      else
                      begin
                        Z[g1,1]:=Z[0,g+3]-Z[0,g+1] ;//a
                        Z[g1,2]:=Z[0,g]-Z[0,g+2] ;   //b
                        Z[g1,3]:=Z[0,g+1]*(Z[0,g+2]-Z[0,g])-Z[0,g]*(Z[0,g+3]-Z[0,g+1]) ;      //c
        if Z[0,g]<Z[0,g+2] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                        else  if Z[0,g]>Z[0,g+2] then Z[g1,0]:=-1; //знак (в какую сторону вектор)
                        if Z[0,g]=Z[0,g+2] then
                          if Z[0,g]<Z[0,g+4] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                          else   if Z[0,g]>Z[0,g+4]  then  Z[g1,0]:=-1;  //знак  (в  какую  сторону
вектор)
        end;
                      //ShowMessage('прямая-'+FloatToStr(g1)+'          a='+FloatToStr(Z[g1,1])+'
b='+FloatToStr(Z[g1,2])+' c='+FloatToStr(Z[g1,3]));
        end;
                  //поиск минимума максимума новых координат фигуры
                  //ShowMessage('поиск минимума максимума новых координат фигуры');
                  Xmin:=Z[0][1];
                  Xmax:=Z[0][1];
                  Ymin:=Z[0][2];
        Ymax:=Z[0][2];
                  For g:=3 to Trunc(F[i,0]*2) do
                  begin
                    if g mod 2<>0 then
                    begin
        if  Z[0][g]<=Xmin  then  Xmin:=Z[0][g];//НЕ  СЧИТАЕТ,  СОХРАНЯЕТ  ЗН.=  0
(если начинать с 0,0 то нормально)
        if Z[0][g]>Xmax then  Xmax:=Z[0][g];
                    end
                    // Поиск Ymin,Ymax
                    else
                    begin
        if  Z[0][g]<=Ymin  then  Ymin:=Z[0][g];//НЕ  СЧИТАЕТ,  СОХРАНЯЕТ  ЗН.=  0
(если начинать с 0,0 то нормально)
        if Z[0][g]>Ymax then  Ymax:=Z[0][g];
        end;
                  end;
                  //поиск матрицы фигуры по найденным координатам
```

```
                //ShowMessage('Угол:'+FloatToStr(a1)+'>>>>>>>>>>
Xmin=('+FloatToStr(Xmin)+'),      Xmax=('+FloatToStr(Xmax)+'Ymin=('+FloatToStr(Ymin)+'),
Ymax=('+FloatToStr(Ymax));
        //онулениематрицышаблона
                For y2:=Trunc(Ymin) to Trunc(Ymax) do
                  For x2:=Trunc(Xmin) to Trunc(Xmax) do
        MatrSH[y2][x2]:=0;
                //поиск матрицы фигуры по найденным координатам
                //сохраняем матрицу шаблона в MatrSH[y2][x2]
        V:=false;
                    for y2:=Trunc(Ymin) to Trunc(Ymax) do
                    begin
                      g:=0;//сколько 1 встроке
                      for x2:=Trunc(Xmin) to Trunc(Xmax) do
        begin
                        V:=false;
                        //перебор прямых, посмотреть сколько пересечений,
                        for g2:=1 to Trunc(F[i,0]) do    //проверка лежит ли точка на осях
шаблона
        begin
                         j45:=Trunc(x2*Z[g2,1]+y2*Z[g2,2]+Z[g2,3]);
                        //ShowMessage('точка  '+FloatToStr(x2)+' , '+FloatToStr(y2)+'прямая
'+FloatToStr(g2)+'>>>>>должнобыть>=0        >>>>>>>>>>        '+FloatToStr(j45));
//+FloatToStr(x2)+'*'+FloatToStr(Z[g2,1])+'+'+FloatToStr(y2)+'*'+FloatToStr(Z[g2,2])+'+'+Flo
atToStr(Z[g2,3])+'='+FloatToStr(j45));
                        if                                                    (j45=0)//and
(((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1])))
//or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1]))
                        then
                        begin
                          if
((g2=Trunc(F[i,0]))and((y2>=Z[0,2])and(y2<=Z[0,g2*2])and(x2>=Z[0,1])and(x2<=Z[0,g2*2-
1])))
                          then
                          begin
                            //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')     >>>>>>>
'+'конечнаяпрямая');
                            MatrSH[y2][x2]:=1; V:=true;  break;
                          end
                          else
                            if                  ((Z[g2,1]=0)and(Z[g2,3]=0))                then
//еслипрямаяпараллельнаосиX
                            begin
                              if (Z[0,g2*2-1]>Z[0,g2*2+1])and(g2<>1)then
                                if ((x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1]))
                                then
                                begin
                                  //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')
>>>>>>>  '+'прямаяПАРАЛперевернутая');
                                  MatrSH[y2][x2]:=1; V:=true;  break;
                                end
                                else if (Z[0,g2*2-1]<Z[0,g2*2+1])and(g2<>1)then
```

```
                                if ((x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1]))
                                then
                                begin
                                  //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')
>>>>>>>  '+'прямаяПАРАЛнормальная');
                                   MatrSH[y2][x2]:=1; V:=true;  break;
        end;
                              end
                              else //если не паралельная и не конечная то
        if((Z[g2,1]<>0)and(Z[g2,3]<>0))and((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z
[0,g2*2-
1])and(x2<=Z[0,g2*2+1]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-
1])and(x2>=Z[0,g2*2+1]))
                               then
                               begin
                               //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')  >>>>>>>
'+'простопрямая');
                               MatrSH[y2][x2]:=1; V:=true;  break;
                               end;
                            end;  //
                          end;
                          if (V=false) then
                          begin
                             j:=0;
                             SetLength(coord,Trunc(F[j,0])+100);//выделениепамяти
                             //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')      >>>>>>>
'+'точкавнепаралибоковых');
        for g2:=1 to Trunc(F[i,0]) do  //проверка лежит ли точка на осях шаблона
        begin
                              if
(((Z[g2,1]<>0)and(Z[g2,2]<>0))or((Z[g2,1]<>0)and(Z[g2,2]=0)))and(((y2>=Z[0,g2*2])and(y2<
=Z[0,g2*2+2]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2]))) then  //
                              begin
                                if  ((g2=Trunc(F[i,0]))and((Z[0,2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,1])<>0))   or   ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])<>0))or ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])=0))
                               then
                               begin
                                 x1:=Trunc((Z[g2,2]*y2+Z[g2,3])/(-Z[g2,1]));
                                 //ShowMessage('(T.   перес)x1='+FloatToStr(x1)+'          (Расм.
T.)x2='+FloatToStr(x2)+'  >>>>>>> if x1<=x2 then j:=j+1;');
                                  if x1<=x2 then
                              begin
                                if j=0 then
        begin
                                  j:=j+1;//сколько точек пересечения до прямой x=0
        coord[j]:=x1;
                               end
                               else
                               begin
                                 cr1:=true;
```

```
                                for cr:=1 to j do
                                  if coord[cr]=x1 then begin cr1:=false; break; end;
        if cr1=true then
                                begin
                                  j:=j+1;//сколько точек пересечения до прямой x=0
        coord[j]:=x1;
                                end;
                               end;
                              end;
                               end;
                              end;
                             end;
                             if j mod 2<>0 then MatrSH[y2][x2]:=1  //еслинечетное
                             else MatrSH[y2][x2]:=0;  //есличетное
                             //ShowMessage('пересекаетдоточких='+FloatToStr(x2)+'
'+FloatToStr(j)+' раз');
                           end;  //
                           //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')          >>>>>>>
'+FloatToStr(Matr[y2][x2]));
                         end;   //x
                        end;    //y
                    //ShowMessage('Угол:'+FloatToStr(a1)+'>>>>>>>>>>фигурасточки
('+FloatToStr(Xmin+k2)+','+FloatToStr(Ymin+k1)+')');
        //наложение матрицы фигуры на матрицу ДО
                    //ShowMessage('наложение матрицы фигуры на матрицу ДО');
        if
((Xmin+k2)>=0)and((Xmax+k2)<=Trunc(Data[4]))and((Ymin+k1)>=0)and((Ymax+k1)<=Trun
c(Data[5]))
                    then
                    begin
                      //ShowMessage('проверяемфигуру, вдиапазонвходит');
                      V:=true;
                      for g:=Trunc(Ymin) to Trunc(Ymax) do
                      begin
                        for g1:=Trunc(Xmin) to Trunc(Xmax) do
                          if  MatrSH[g][g1]=1  then   if  MatrSH[g][g1]<>Matr0[g+k1][g1+k2]  then
begin V:=false;  break; end;     //
                        if V=false then break;
                      end;
                    end
                    else V:=false;
                    //ShowMessage('V:= '+BoolToStr(V));
        if V=true then
                    begin
                      //онудение матрицы если фигура входит
        for g:=Trunc(Ymin) to Trunc(Ymax) do
                        for g1:=Trunc(Xmin) to Trunc(Xmax) do
                          if MatrSH[g][g1]=1 then Matr0[g+k1][g1+k2]:=0;
                      k:=k+1;
                      Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
                      Frame101.Label4.Caption:='Вмещениешаблона     '+FloatToStr(i)+'     вида.
Вместили '+FloatToStr(k)+' шт.';
```

```
                    Frame101.Label4.Refresh;
                    //ShowMessage(FloatToStr(k));
                    Matr0[k1][k2]:=1;
        break;//прерывание поиска угла
                  end;
                end;//КОНЕЦ действие если разрешение:=true;
        if  DataF[n][8]<>2 then
                  if
((Frame31.LabeledEdit1.Text<>")and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                  else yg:=90;
                  if DataF[n][8]=2 then
                  begin
                    if (DataF[t][3]<>DataF[t][5]) then//еслипрямоугольник
                    begin
                      if                Frame31.CheckBox1.Checked                =                false
then//есликлаишазапретаненажатато
                        if
((Frame31.LabeledEdit1.Text<>")and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                        else yg:=90;
                     if Frame31.CheckBox1.Checked = true then yg:=90;
                    end;
                    if (DataF[t][3]=DataF[t][5]) then//если rdf
                    begin
                      if                Frame31.CheckBox1.Checked                =                false
then//есликлаишазапретаненажатато
                        if
((Frame31.LabeledEdit1.Text<>")and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                        else yg:=360;
                     if Frame31.CheckBox1.Checked = true then yg:=360;
                    end;
                  end;
                  a1:=a1-yg;
                until  ((a1=0));
                if ((V=true)) then Matr0[k1][k2]:=0;
              end;
                        end;
            end;
          END;
          Ksh[i]:=k;//ЗАПОЛНЕНИЕМАССИВАСК
          //ShowMessage('всего: '+FloatToStr(k));
          g2A:=0;
          g2B:=0;
          g1:=0;
        end;
        //ShowMessage('Этап 2');
        Frame101.Label3.Caption:=''; Frame101.Label3.Refresh;
```

```
  Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
 Frame101.Label3.Caption:='Перебор всех вариантов карт раскроя. ';
  Frame101.Label3.Refresh;
  //Этап 2
 j3:=0;
  j4:=0;
  //сколько всего шаблонов раскраиваться будет
 for i1:=0 to Ksh[1] do
    for i2:=0 to Ksh[2] do
      for i3:=0 to Ksh[3] do
        for i4:=0 to Ksh[4] do
          for i5:=0 to Ksh[5] do
           if NOT
((((i1>0)and(i1<Ksh[1])and(i2=0)and(i3=0)and(i4=0)and(i5=0))or((i1=0)and(i2>0)and(i2<Ksh[
2])and(i3=0)and(i4=0)and(i5=0))or((i1=0)and(i2=0)and(i3>0)and(i3<Ksh[3])and(i4=0)and(i5=0
))or((i1=0)and(i2=0)and(i3=0)and(i4<>0)and(i4<Ksh[4])and(i5=0))or((i1=0)and(i2=0)and(i3=0)
and(i4=0)and(i5>0)and(i5<Ksh[5])))  )
            or (not ((i1+i2+i3+i4+i5)=0) and not
((DataF[1][1]*i1+DataF[2][1]*i2+DataF[3][1]*i3+DataF[4][1]*i4+DataF[5][1]*i5)>Data[2]))
 then
            begin
             j3:=j3+1;  //сколько МАКСИМУМ вариантов раскроя для освободжение
памяти для  STabl2
 j4:=j4+i1+i2+i3+i4+i5;//сколькомаксимумдля Raskroi2
            end;

  //ShowMessage('МАКСИМУМвариантовраскроя'+FloatToStr(j3)); //t:=0//
  if j3>0 then t29:=850/(j3);
  if shabl=1 then t29:=850;
 //освобождение матрицы
  SetLength(STabl2,(100),(j3));//выделение области памяти для матрицы  (у или
строка макс, х или столбец максимально)
 SetLength(Raskroi2,(j4+1000),(100));
  g2A:=0;
  g2B:=0;
  i:=1;
  rs:=1;
  For t:=0 to (Trunc(Data[5])) do
    For t1:=0 to (Trunc(Data[4])) do
      Matr0[t][t1]:=Matr[t][t1];
 //ДАЛЬНЕЙШЕЕ ЗАПОЛНЕНИЕ МАТРИЦЫ С НЕСКОЛЬКИМИ ШАБЛОНАМИ
НА ДО
 for i1:=0 to Ksh[1] do
    for i2:=0 to Ksh[2] do
      for i3:=0 to Ksh[3] do
        for i4:=0 to Ksh[4] do
 for i5:=0 to Ksh[5] do  //если все ок то добавить до 15 вариантов шаблонов
 {for i1:=Ksh[1] Downto 0 do
    for i2:=Ksh[2] Downto 0 do
      for i3:=Ksh[3] Downto 0 do
        for i4:=Ksh[4] Downto 0 do
 for i5:=Ksh[5] Downto 0 do  }//если все ок то добавить до 15 вариантов шаблонов
```

```
begin
        Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
        Frame101.Label4.Caption:='Картараскроя № '+FloatToStr(i);
Frame101.Label4.Refresh;
        //не отображается раскрой в котором учавствует 1 не нулевой элемент, и при таком условии раскраивается только элемент равный максимальному кол-ву
if
Not(((i1>0)and(i1<Ksh[1])and(i2=0)and(i3=0)and(i4=0)and(i5=0))or((i1=0)and(i2>0)and(i2<Ksh[2])and(i3=0)and(i4=0)and(i5=0))or((i1=0)and(i2=0)and(i3>0)and(i3<Ksh[3])and(i4=0)and(i5=0))or((i1=0)and(i2=0)and(i3=0)and(i4<>0)and(i4<Ksh[4])and(i5=0))or((i1=0)and(i2=0)and(i3=0)and(i4=0)and(i5>0)and(i5<Ksh[5]))) then
begin
        //обнуление расматриваемого варианта
        for t:=1 to 8 do STabl2[t,i]:=0;
//первонвидматрицы
        For t:=0 to (Trunc(Data[5])) do
          For t1:=0 to (Trunc(Data[4])) do
            Matr0[t][t1]:=Matr[t][t1];
        //обнулениематриц  Z[g,0...100]  for g:=0 to 100 do
        For t:=0 to 50 do
          For t1:=0 to 50 do
            Z[t,t1]:=0;
        for t:=1 to 5 do L[t]:=0;
        if (i1+i2+i3+i4+i5)<>0 then
begin
         if
(DataF[1][1]*i1+DataF[2][1]*i2+DataF[3][1]*i3+DataF[4][1]*i4+DataF[5][1]*i5)<=Data[2] then   //НО сравнение не только по площадям, но и по сторонам, так как необходимо все таки отобранные варианты проверять насамом деле ли они войдут на лист или лажа
          begin
           //дублирование кол Ii в L[i]
L[1]:=i1;
           L[2]:=i2;
           L[3]:=i3;
           L[4]:=i4;
           L[5]:=i5;
           k:=0;
           //ShowMessage('рассматриваетсяраскрой('+FloatToStr(i)+')');
           For k1:=(0) to Trunc(Data[5]) do//Y   //переборданныхДО
           BEGIN
                                                    for t:=1 to 5 do
                                                        if L[t]>0 then
begin
                    //сохранение сторон прямоугольника шаблона для дальнейшей работы
for k2:=(0) to Trunc(Data[4]) do //X
                    begin
                      if Matr0[k1][k2]=0 then
                      begin

Frame101.ProgressBar1.Position:=Trunc((t29/((i1+i2+i3+i4+i5)*Data[4]*Data[5]))+t30);
                        t30:=(t29/((i1+i2+i3+i4+i5)*Data[4]*Data[5]))+t30;
```

```
Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
                          Frame101.Label5.Refresh;
                        end;
                        //ShowMessage(FloatToStr(i)+'                >>>>>                Matr0
('+FloatToStr(k2)+','+FloatToStr(k1)+')=' +FloatToStr(Matr0[k1][k2]));
                        if Matr0[k1][k2]=1 then
                        begin
                          a1:=360;
                          f11:=False;
                          repeat
                            razrehenie:=true;
                            if ((F11=true)) then
                            begin
                              g1:=1;
                              //отражениефигуры
                              for g:=Trunc(F[t,0]*2) downto 1 do
                                if g mod 2=0 then  F1[g1+1]:=F[t,g]        //если y
                                else
                                begin
                                  F1[g1]:=F[t,g]-2*F[t,g]+DataF[t][3];
        g1:=g1+2;
                                  //вывод координат для проверки
        //ShowMessage('1. координаты: '+FloatToStr(Z[0,g])+';'+FloatToStr(Z[0,g+1]));
        end;  //если x
                              //сохранили коорлинаты нового положения фигуры
        for g:=1 to Trunc(F[t,0]*2) do
                              ifgmod 2<>0 then//еслих
        begin//по математическим правилам до ближайшего целого:
        Z[0,g]:=Round(F1[g]*cos(a1*3.14/180)-F1[g+1]*sin(a1*3.14/180));//x   //k2

Z[0,g+1]:=Round(F1[g+1]*cos(a1*3.14/180)+F1[g]*sin(a1*3.14/180));//y //k1
        //добавить к координатам текущее положение
                              //если хотя бы одна новая координата не входит в ДО то
прерывания поиска
        //разрешение:=true;
                               k9:=Round(Z[0,g]+k2); //x
                              k10:=Round(Z[0,g+1]+k1); //y
                              if NOT ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and
(k9<=Data[4])))) then begin razrehenie:=false; break; end;
                              if ((((k9>0) and (k10>0))and((k10<Data[5]) and (k9<Data[4]))))
then
                                if Matr0[k10][k9]=0 then begin razrehenie:=false; break; end;
                              //ShowMessage('1.                                    координаты:
'+FloatToStr(Z[0,g])+';'+FloatToStr(Z[0,g+1]));
        end;
                            end
                            else//сохранили коорлинаты нового положения фигуры без
отражения//сохранили коорлинаты нового положения фигуры
        for g:=1 to Trunc(F[t,0]*2) do
                              if g mod 2<>0 then//еслих
        begin
```

```
//// по математическим правилам до ближайшего целого:
Z[0,g]:=Round(F[t,g]*cos(a1*3.14/180)-F[t,g+1]*sin(a1*3.14/180));//x

Z[0,g+1]:=Round(F[t,g+1]*cos(a1*3.14/180)+F[t,g]*sin(a1*3.14/180));//y
//добавить к координатам текущее положение
                        //если хотя бы одна новая координата не входит в ДО то
прерывания поиска
//разрешение:=true;
                        k9:=Round(Z[0,g]+k2); //x
                        k10:=Round(Z[0,g+1]+k1); //y
                        if NOT ((((k9>=0) and (k10>=0))and((k10<=Data[5]) and
(k9<=Data[4])))) then begin razrehenie:=false; break; end;
                        if ((((k9>0) and (k10>0))and((k10<Data[5]) and (k9<Data[4]))))
then
                        if Matr0[k10][k9]=0 then begin razrehenie:=false; break; end;
                        //ShowMessage('1.                                    координаты:
'+FloatToStr(Z[0,g])+';'+FloatToStr(Z[0,g+1]));
                      end;
                    if razrehenie=true then
                    begin
g1:=0;//прямых
                     //заполнение уравнений прямых в фигуре
for g:=1 to Trunc(F[t,0]*2) do
                      if g mod 2<>0 then//еслих
                      begin
                        g1:=g1+1;
                        if g=F[t,0]*2-1 then
                        begin
                          Z[g1,1]:=Z[0,2]-Z[0,g+1] ;//a
                          Z[g1,2]:=Z[0,g]-Z[0,1] ;  //b
                          Z[g1,3]:=Z[0,g+1]*(Z[0,1]-Z[0,g])-Z[0,g]*(Z[0,2]-Z[0,g+1])      ;
//c
if Z[0,1]<Z[0,3] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                          else  if Z[0,1]>Z[0,3] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
                          if Z[0,1]=Z[0,3] then
                            if Z[0,1]<Z[0,5] then Z[g1,0]:=1 //знак (в какую сторону
вектор)
                            else    if Z[0,1]>Z[0,5] then Z[g1,0]:=-1; //знак (в какую
сторону вектор)
end
                      else
                      begin
                        Z[g1,1]:=Z[0,g+3]-Z[0,g+1] ;//a
                        Z[g1,2]:=Z[0,g]-Z[0,g+2] ;  //b
                        Z[g1,3]:=Z[0,g+1]*(Z[0,g+2]-Z[0,g])-Z[0,g]*(Z[0,g+3]-
Z[0,g+1]) ;     //c
if Z[0,g]<Z[0,g+2] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                        else    if Z[0,g]>Z[0,g+2] then Z[g1,0]:=-1; //знак (в какую
сторону вектор)
                        if Z[0,g]=Z[0,g+2] then
```

```
                                if Z[0,g]<Z[0,g+4] then Z[g1,0]:=1 //знак (в какую сторону
вектор)
                                else  if Z[0,g]>Z[0,g+4] then Z[g1,0]:=-1; //знак (в какую
сторону вектор)
        end;
                                  //ShowMessage('прямая-'+FloatToStr(g1)+'
a='+FloatToStr(Z[g1,1])+' b='+FloatToStr(Z[g1,2])+' c='+FloatToStr(Z[g1,3]));
        end;
                              //поиск минимума максимума новых координат фигуры
                              Xmin:=Z[0][1];
                              Xmax:=Z[0][1];
                              Ymin:=Z[0][2];
        Ymax:=Z[0][2];
                              For g:=3 to Trunc(F[t,0]*2) do
                              begin
                                if g mod 2<>0 then
                                begin
        if Z[0][g]<=Xmin then Xmin:=Z[0][g];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
        if Z[0][g]>Xmax then  Xmax:=Z[0][g];
                                end
                                // Поиск Ymin,Ymax
                                else
                                begin
        if Z[0][g]<=Ymin then Ymin:=Z[0][g];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
        if Z[0][g]>Ymax then  Ymax:=Z[0][g];
                                end;
                              end;
                              //ShowMessage('Xmin='+FloatToStr(Xmin)+'
Xmax='+FloatToStr(Xmax)+'    Ymin='+FloatToStr(Ymin)+' Ymax='+FloatToStr(Ymax));
        //поиск матрицы фигуры по найденным координатам
                              //ShowMessage('поиск матрицы фигуры по найденным
координатам');
        //онулениематрицышаблона
                              For y2:=Trunc(Ymin) to Trunc(Ymax) do
                                For x2:=Trunc(Xmin) to Trunc(Xmax) do
        MatrSH[y2][x2]:=0;
                              //поиск матрицы фигуры по найденным координатам
                              //сохраняем матрицу шаблона в MatrSH[y2][x2]
        V:=false;
                              for y2:=Trunc(Ymin) to Trunc(Ymax) do
                              begin
                                g:=0;//сколько 1 встроке
                                for x2:=Trunc(Xmin) to Trunc(Xmax) do
        begin
                                  V:=false;
                                  //перебор прямых, посмотреть сколько пересечений,
                                  for g2:=1 to Trunc(F[t,0]) do   //проверка лежит ли точка на осях
шаблона
        begin
                                   j45:=Trunc(x2*Z[g2,1]+y2*Z[g2,2]+Z[g2,3]);
```

```
                                //ShowMessage('точка '+FloatToStr(x2)+' ,
'+FloatToStr(y2)+'прямая '+FloatToStr(g2)+'>>>>>должнобыть>=0 >>>>>>>>>
'+FloatToStr(j45));
//+FloatToStr(x2)+'*'+FloatToStr(Z[g2,1])+'+'+FloatToStr(y2)+'*'+FloatToStr(Z[g2,2])+'+'+Flo
atToStr(Z[g2,3])+'='+FloatToStr(j45));
                                if (j45=0)//and
(((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1])))
//or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1]))
                                then
                                begin
                                  if
((g2=Trunc(F[t,0]))and((y2>=Z[0,2])and(y2<=Z[0,g2*2])and(x2>=Z[0,1])and(x2<=Z[0,g2*2-
1])))  then begin
                                  //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')
>>>>>>>  '+'конечнаяпрямая');
                                  MatrSH[y2][x2]:=1; V:=true;  break; end
        else
                                  if  ((Z[g2,1]=0)and(Z[g2,3]=0))  then  //если  прямая
параллельна оси X
        begin
                                  if  (Z[0,g2*2-1]>Z[0,g2*2+1])and(g2<>1)then  if
((x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1])) then begin
                                  //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')
>>>>>>>  '+'прямаяПАРАЛперевернутая');
                                   MatrSH[y2][x2]:=1; V:=true;  break; end
                                  else  if  (Z[0,g2*2-1]<Z[0,g2*2+1])and(g2<>1)then  if
((x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1])) then begin
                                  //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')
>>>>>>>  '+'прямаяПАРАЛнормальная');
                                  MatrSH[y2][x2]:=1; V:=true;  break; end;
        end
                                 else //если не паралельная и не конечная то
        if((Z[g2,1]<>0)and(Z[g2,3]<>0))and((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z
[0,g2*2-
1])and(x2<=Z[0,g2*2+1]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-
1])and(x2>=Z[0,g2*2+1]))
                                  then
                                  begin
                                    //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')
>>>>>>>  '+'простопрямая');
                                    MatrSH[y2][x2]:=1;
                                    V:=true;
                                    break;
                                  end;
                                end;
                              end;
                              if (V=false) then
                              begin
                                j:=0;
                                SetLength(coord,Trunc(F[j,0])+100);//выделениепамяти
                                //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')
>>>>>>>  '+'точкавнепаралибоковых');
```

```
        for g2:=1 to Trunc(F[t,0]) do  //проверка лежит ли точка на осях шаблона
        begin
                                    if
(((Z[g2,1]<>0)and(Z[g2,2]<>0))or((Z[g2,1]<>0)and(Z[g2,2]=0)))and(((y2>=Z[0,g2*2])and(y2<
=Z[0,g2*2+2]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2]))) then  //
                                    begin
                                      if                              ((g2=Trunc(F[t,0]))and((Z[0,2]-
Z[0,g2*2])<>0)and((Z[0,g2*2-1]-Z[0,1])<>0))     or     ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-
Z[0,g2*2])<>0)and((Z[0,g2*2-1]-Z[0,g2*2+1])<>0))or ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-
Z[0,g2*2])<>0)and((Z[0,g2*2-1]-Z[0,g2*2+1])=0))
                                      then
                                      begin
                                        x1:=Trunc((Z[g2,2]*y2+Z[g2,3])/(-Z[g2,1]));
                                        //ShowMessage('(T. перес)x1='+FloatToStr(x1)+'      (Расм.
T.)x2='+FloatToStr(x2)+'  >>>>>>> if x1<=x2 then j:=j+1;');
                                        if x1<=x2 then
                                  begin
                                   if j=0 then
        begin
                                     j:=j+1;//сколько точек пересечения до прямой x=0
        coord[j]:=x1;
                                   end
                                   else
                                   begin
                                     cr1:=true;
                                     for cr:=1 to j do
                                       if coord[cr]=x1 then begin cr1:=false; break; end;
        if cr1=true then
                                     begin
                                       j:=j+1;//сколько точек пересечения до прямой x=0
        coord[j]:=x1;
                                     end;
                                   end;
                                  end;
                                            end;
                                       end;
                                     end;
                                     if j mod 2<>0 then MatrSH[y2][x2]:=1  //еслинечетное
                                     else MatrSH[y2][x2]:=0;  //есличетное
                                     //ShowMessage('пересекаетдоточких='+FloatToStr(x2)+'
'+FloatToStr(j)+' раз');
                                     end;
                                    //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')
>>>>>>>  '+FloatToStr(Matr[y2][x2]));
        end;   //x
                                  end;   //y
                                  //наложение матрицы фигуры на матрицу ДО
                                  //ShowMessage('наложение матрицы фигуры на матрицу ДО');
        if
((Xmin+k2)>=0)and((Xmax+k2)<=Trunc(Data[4]))and((Ymin+k1)>=0)and((Ymax+k1)<=Trun
c(Data[5]))
                                  then
```

```
                                begin
                                  //ShowMessage('проверяемфигуру, вдиапазонвходит');
                                  V:=true;
                                  for g:=Trunc(Ymin) to Trunc(Ymax) do
                                  begin
                                    for g1:=Trunc(Xmin) to Trunc(Xmax) do
                                      if MatrSH[g][g1]=1 then if
MatrSH[g][g1]<>Matr0[g+k1][g1+k2] then  begin V:=false;  break; end;     //
                                    if V=false then break;
                                  end;
                                end
                                else V:=false;
                                //ShowMessage('V:= '+BoolToStr(V));
                                if V=true then
                                begin
                                  //ShowMessage('онудениематрицыеслифигуравходит');
                                  for g:=Trunc(Ymin) to Trunc(Ymax) do
                                    for g1:=Trunc(Xmin) to Trunc(Xmax) do
                                      if MatrSH[g][g1]=1 then Matr0[g+k1][g1+k2]:=0;
                                  Matr0[k1][k2]:=1;
                                  L[t]:=L[t]-1;
                                  //ShowMessage('вписали '+FloatToStr(k)+'; осталось
'+FloatToStr(L[t]));
        //сохранение координат
                                  k:=k+1;
                                  Raskroi2[k+rs-1,2]:=t;//сохранение номера шаблона
        Raskroi2[k+rs-1,3]:=k2;//x0
                                  Raskroi2[k+rs-1,4]:=k1;//y0
        Raskroi2[k+rs-1,5]:=a1;//ориентация(угол поворота относительно x0y0 против
часовой стрелки
                                  break;//прерывание поиска угла
                                end;
                              end;//КОНЕЦ действие если разрешение:=true;
        if  DataF[t][8]<>2 then
                                if
((Frame31.LabeledEdit1.Text<>")and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                                  else yg:=90;
                                if DataF[t][8]=2 then
                                begin
                                  if (DataF[t][3]<>DataF[t][5]) then//еслипрямоугольник
                                  begin
                                    if Frame31.CheckBox1.Checked = false
then//ecликлаишазапретаненажатато
                                      if
((Frame31.LabeledEdit1.Text<>")and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                                    else yg:=90;

                                  if Frame31.CheckBox1.Checked = true then yg:=90;
```

```
                                end;
                                if (DataF[t][3]=DataF[t][5]) then//если rdf
                                begin
                                  if Frame31.CheckBox1.Checked = false
then//есликлаишазапретаненажатато
                                    if
((Frame31.LabeledEdit1.Text<>")and(StrToInt(Frame31.LabeledEdit1.Text)<360)and(StrToInt(
Frame31.LabeledEdit1.Text)>=0)) then yg:=StrToInt(Frame31.LabeledEdit1.Text)//45;//yg:= 1;
//
                                     else yg:=360;
                                  if Frame31.CheckBox1.Checked = true then yg:=360;
                                end;
                              end;

Frame101.ProgressBar1.Position:=Trunc((t29/((i1+i2+i3+i4+i5)*Data[4]*Data[5]*(360/yg)))+t3
0);
                                t30:=(t29/((i1+i2+i3+i4+i5)*Data[4]*Data[5]*(360/yg)))+t30;

Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
                                Frame101.Label5.Refresh;
                                a1:=a1-yg;
                                //отображение
                                //if ((a1=0)and(F11=false)and(DataF[n][8]<>2)) then begin a1:=360;
F11:=true; end; // поворотфигуры
                              until ((a1=0));
                              if ((V=true)) then Matr0[k1][k2]:=0;
        end;//если =1
                            //если данный вид шаблона уже включен то убираем его из
расмотрения
                            if L[t]<=0 then break;//если вставили нужное кол-во шаблонов то
остановка
        end;  //k2

                          end; //L[t]
                        end;//k1
        //проверка, все ли шаблоны вошли в ДО
        t1:=0;
                        for t:=1 to 5 do if L[t]>0 then t1:=t1+1;
        if t1>0 then  //если в L[t] есть не нулевый элементы, значит что эти элементы не
вошли //удаляем данный раскрой из таблицы
                        begin //то вошли не все, раскрой отменяется
        //ShowMessage('Отмена');
                         Frame101.ProgressBar1.Position:=Trunc((t29)+t30);
                         t30:=((t29)+t30);

Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
        Frame101.Label5.Refresh;
                         //первоначальный вид матрицы
        For t:=0 to (Trunc(Data[5])) do
                           For t1:=0 to (Trunc(Data[4])) do
                             Matr0[t][t1]:=Matr[t][t1];
                        //обнулениераскр2
```

```
                    For t:=rs to (rs+k-1) do
////////////////////////////////////////////////////////////////////////////////////////
                     For t1:=1 to (5) do
                      Raskroi2[t,t1]:=0;
        //обнуление 2
                    k:=0;
                   end//отмена раскроя
                   else//применение раскроя
        begin
                    Frame101.ProgressBar1.Position:=Trunc((t29)+t30);
                    t30:=((t29)+t30);

Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
                    Frame101.Label5.Refresh;
                    Frame101.Label4.Caption:=''; Frame101.Label4.Refresh;
                    Frame101.Label4.Caption:='Картараскроя № '+FloatToStr(i)+'.
Сохранениеданныхраскроя.';
                    Frame101.Label4.Refresh;
                    //ShowMessage('Сохранение');
                    {e1:=i1;e2:=i2;e3:=i3;e4:=i4;e5:=i5;}
        // Создать объект автоматизации KOMPAS_Graphic  ранее
                    //Получаем и заполняем интерфейс параметров документа
        DocumentParam:=ksDocumentParam(kompas.GetParamStruct(ko_DocumentParam));
                    DocumentParam.Init();
                    DocumentParam.type_:=ksDocumentFragment;//Drawing;
                    //Получаеминтерфейсдокумента
                    Doc2:=ksDocument2D(kompas.Document2D);
                    //Создаемновыйчертеж
                    Doc2.ksCreateDocument(DocumentParam);
                    //ShowMessage('раскрой'+FloatToStr(i)+' всегошаблонов'+FloatToStr(k));
                    Raskroi2[rs,1]:=k;
                    STabl2[1,i]:=1;  //перебор № ДО
                    STabl2[2,i]:=i;  //перебор № листа
                    STabl2[3,i]:=i1;
                    STabl2[4,i]:=i2;
                    STabl2[5,i]:=i3;
                    STabl2[6,i]:=i4;
                    STabl2[7,i]:=i5;
                    STabl2[8,i]:=Data[2]-
(DataF[1][1]*i1+DataF[2][1]*i2+DataF[3][1]*i3+DataF[4][1]*i4+DataF[5][1]*i5);//остатокотр
аскроя (поПРЯМОУГОЛЬНИКАМ)
                    //рисование
                    Frame31.Image2.Canvas.Brush.Style:=bsSolid;
                    Frame31.Image2.Canvas.Brush.Color:=clWhite;//$00F5F7F9;
                    Frame31.Image2.Canvas.rectangle(0,0,width,height);
                    Frame31.Image2.Width:=500;//325;//пох
                    Frame31.Image2.Height:=500;//425;//по y
                    Frame31.Image2.Canvas.pen.Color:=clBlack;//$00F5F7F9;
                    Frame31.Image2.Canvas.MoveTo(0,0);     //по x
                    Frame31.Image2.Canvas.LineTo(Width,0);
                    Frame31.Image2.Canvas.MoveTo(0,0);  //по y
                    Frame31.Image2.Canvas.LineTo(0,Height);
```

```
                    if Data[4]>Data[5] then m:=Trunc(500/Data[4]) else
m:=Trunc(500/Data[5]);
                    u:=0;
                                    For u2:=6 to (Trunc(Data[0])*2+5)
do//переборкоординат
                                 if u2 mod 2=0 then
                                        begin
                                          xu:=Trunc(Data[u2])*m+2;//том на 100 убрать тут
чтоб видно было тк размер примера мал
                                 yu:=Trunc(Data[u2+1])*m+2;
                                          PointArray[u]:=Point(xu,yu); u:=u+1;
                        Frame31.Image2.Canvas.MoveTo(xu,0);  //по y
                        Frame31.Image2.Canvas.LineTo(xu,5);
                        Frame31.Image2.Canvas.MoveTo(0,yu);  //по y
                        Frame31.Image2.Canvas.LineTo(5,yu);
                        //рисованиевкомпас
                        if  u2<(Trunc(Data[0])*2+5-1) then Doc2.ksLineSeg(  Trunc(Data[u2]),
Trunc(Data[u2+1]), Trunc(Data[u2+2]),  Trunc(Data[u2+3]), 1 );
                        if  u2=(Trunc(Data[0])*2+5-1) then Doc2.ksLineSeg(  Trunc(Data[u2]),
Trunc(Data[u2+1]), Trunc(Data[6]),  Trunc(Data[7]), 1 );
                                    end;

Frame31.Image2.Canvas.pen.Color:=$00E4E4E4;//clBlack;//clSilver;//фонДО

Frame31.Image2.Canvas.Brush.Color:=$00E4E4E4;//clBlack;//clSilver;//фонДО

Frame31.Image2.Canvas.Polygon(Slice(PointArray,u));//прорисовкаполигона
                    //ShowMessage('Раскрой                                    '+FloatToStr(i)+':
шаблоноввсего'+FloatToStr(Raskroi[rs][1]));
                    u:=0;
                                    //рисованиешаблонов
                                    For u2:=(rs) to (Trunc(Raskroi2[rs][1])+rs-1) do
                                    begin
                     t:=Trunc(Raskroi2[u2][2]);//номершаблона
                     a1:=Trunc(Raskroi2[u2][5]);//угол
                     for u3:=1 to Trunc(F[t,0]*2) do
                      if u3 mod 2<>0 then//еслих
        begin
        //// по математическим правилам до ближайшего целого:
        xu:=Round(((F[t,u3]*cos(a1*3.14/180)-
F[t,u3+1]*sin(a1*3.14/180))+Raskroi2[u2][3]))*m+2;//x

yu:=Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi2[u2][4]))*m+2;//
y
                        Frame31.Image2.Canvas.MoveTo(xu,0);  //по y
                        Frame31.Image2.Canvas.LineTo(xu,5);
                        Frame31.Image2.Canvas.MoveTo(0,yu);  //по y
                        Frame31.Image2.Canvas.LineTo(5,yu);
                        //ShowMessage(FloatToStr(u2)+'                                    координаты:
'+FloatToStr(xu)+';'+FloatToStr(yu));
                                          PointArray[u]:=Point(xu,yu); u:=u+1;
                        //рисованиевкомпас
```

```
                    if u3<Trunc(F[t,0]*2-1) then Doc2.ksLineSeg(
Round(((F[t,u3]*cos(a1*3.14/180)-F[t,u3+1]*sin(a1*3.14/180))+Raskroi2[u2][3])),
Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi2[u2][4])),Round(((F[t,
u3+2]*cos(a1*3.14/180)-F[t,u3+3]*sin(a1*3.14/180))+Raskroi2[u2][3])),
Round(((F[t,u3+3]*cos(a1*3.14/180)+F[t,u3+2]*sin(a1*3.14/180))+Raskroi2[u2][4])), 1 );
                    if u3=Trunc(F[t,0]*2-1) then Doc2.ksLineSeg(
Round(((F[t,u3]*cos(a1*3.14/180)-F[t,u3+1]*sin(a1*3.14/180))+Raskroi2[u2][3])),
Round(((F[t,u3+1]*cos(a1*3.14/180)+F[t,u3]*sin(a1*3.14/180))+Raskroi2[u2][4])),Round(((F[t,
1]*cos(a1*3.14/180)-F[t,2]*sin(a1*3.14/180))+Raskroi2[u2][3])),
Round(((F[t,2]*cos(a1*3.14/180)+F[t,1]*sin(a1*3.14/180))+Raskroi2[u2][4])), 1 );
                     end;

Frame31.Image2.Canvas.pen.Color:=clBlack;//clSilver;//фонДО$00E4E4E4;//
                    Frame31.Image2.Canvas.Brush.Color:=$00E4E4E4; // Россия

Frame31.Image2.Canvas.pen.Color:=clBlack;//clSilver;//фонДО$00E4E4E4;//
                    case Trunc(Raskroi2[u2][2]) of
                     1:Frame31.Image2.Canvas.Brush.Color:=clGreen; // Россия
                     2:Frame31.Image2.Canvas.Brush.Color:=clYellow; // Англия
                     3:Frame31.Image2.Canvas.Brush.Color:=clRed; // Австрия
                     4:Frame31.Image2.Canvas.Brush.Color:=clBlue; // Германия, Дания,
Исландия, Нидерланды
                     5:Frame31.Image2.Canvas.Brush.Color:=clBlack; // Италия
                    end;

Frame31.Image2.Canvas.Polygon(Slice(PointArray,u));//прорисовкаполигона
                     u:=0;
                    end;
                    Frame31.Image2.Picture.SaveToFile(ProgDIR                    +'\show
projects'+'\'+Frame21.edit1.text+'\'+'Nesting №2'+'\'+IntToStr(i)+'.bmp');
                    Doc2.ksText(0,Trunc(Data[5]+20),0,5,0,0,'Раскрой №'+FloatToStr(i));
                    //заполнениетаблицы
                    for i0:=1 to Shabl do Doc2.ksText(0,(Data[5]+3+(i0-
1)*3),0,2,0,0,'Шаблон № '+FloatToStr(i0)+'; Кол-во: '+FloatToStr(STabl2[i0+2,i]));
                    Doc2.ksText(0,(Data[5]+3+(Shabl)*3),0,2,0,0,'Sост.                =
'+FloatToStr(Trunc(STabl2[8,i]))+' (мм2); Заполнения = '+FloatToStr(Trunc((Data[2]-
STabl2[8,i])*100/Data[2]))+' % ');
                    //6 закрытие 1
                    doc2.ksSaveDocument(ProgDIR                    +'\show
projects'+'\'+Frame21.edit1.text+'\'+'Nesting №2'+'\'+IntToStr(i)+'.cdw');
                    //закрытиефрагмента
                    if doc2<>nil then doc2:=nil;
        i:=i+1;
                    rs:=rs+k+1;
                    k:=0;
                   end;//применение раскроя
                  end;//условие 2
                 end;//условие 1
                end;//конец если1
               end;//варианты
          Raskr2:=rs;//в какой позиции курсор
          Raskr3:=i-1; //сколько всего вариантов раскроя
```

```
razr2:=true;
  Frame31.StringGrid2.RowCount:=Raskr3+1+1;
//создание таблицы отображения данных раскроя
  //ШАПКА
Frame31.StringGrid2.Cells[0, 0]:='№';
  Frame31.StringGrid2.Cells[1, 0]:='ДО';
  for i:=1 to Shabl do Frame31.StringGrid2.Cells[i+1, 0]:=IntToStr(i);  //видшаблона, использовавшегопираскрое;
  Frame31.StringGrid2.Cells[Shabl+2, 0]:='S ост.';  //площальостатка
  Frame31.StringGrid2.Cells[Shabl+3, 0]:='% заполнения';  //площальостатка
  //заполнениеданных
  for i:=1 to Raskr3 do
  begin
    Frame31.StringGrid2.Cells[0, i]:=FloatToStr(STabl2[2,i]);//номервариантараскроя
    Frame31.StringGrid2.Cells[1,  i]:=FloatToStr(STabl2[1,i]);  //видделовогоостатка, использовавшегопираскрое
    for i1:=1 to Shabl do Frame31.StringGrid2.Cells[i1+1, i]:=FloatToStr(STabl2[i1+2,i]); //видшаблона, использовавшегопираскрое  //видшаблона, использовавшегопираскрое;
    Frame31.StringGrid2.Cells[Shabl+2,  i]:=FloatToStr(Trunc(STabl2[8,i])); //площальостатка
    Frame31.StringGrid2.Cells[Shabl+3,  i]:=FloatToStr(Trunc((Data[2]-STabl2[8,i])*100/Data[2]));  //площальостатка
  end;
  Frame31.StringGrid2.Cells[Shabl+2, Raskr3+1]:='Итого:';
  Frame31.StringGrid2.Cells[Shabl+3,  Raskr3+1]:=FloatToStr(Raskr3); //всеговариантовраскроя
  Frame31.PageControl1.Visible:=true;//открываетсяфрейм 3
  SpeedButton3.Enabled:=false;  //раскройзапрещен
  if Raskr3<25 then Frame31.StringGrid2.Height:=(3+Raskr3)*20+5
  else Frame31.StringGrid2.Height:=(1+25)*20;
  Panel1.Enabled:=true;//заблокированныпанелькнопок
  SpeedButton3.Enabled:=False;
  Frame31.Visible:=true;
//видимость фрама отображения анамации
  Frame101.Visible:=false;
if (Raskr3)>=1 then
  begin
    Frame31.Image8.Picture.LoadFromFile(ProgDIR +'\show projects\'+Frame21.edit1.text+'\'+'Nesting №2'+'\'+IntToStr(1)+'.bmp');
    Frame31.Image8.Proportional:=true;
    Frame31.Image8.Transparent:=true;
  end;
  h02:=1;
  h02max:=Raskr3;
  Frame101.ProgressBar1.Position:=1000;
  Frame101.Label5.Caption:=FloatToStr(Frame101.ProgressBar1.Position/10);
  Frame101.Label5.Refresh;
  Frame31.ShellListView4.Root:=ProgDIR +'\show projects'+'\'+Frame21.edit1.text+'\'+'Nesting №2';
//Закрытие компас в общем
  if Kompas <> nil then
  begin
```

```
    //принудительно закрыть
    Kompas.Quit;
Kompas := nil;
end;
end;
```

ПРИЛОЖЕНИЕ П

Листинг кнопки добавления ДО

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  FilePath:string;//имяфайласДО
  m,j,x,y,g,g1,y2,x2,g2,x1:Integer;//nxm-размерматрицы //координаты
  V,F11:Boolean;
  Xmin,Xmax,Ymin,Ymax,j45:real;
Z: array [0..100,0..100] of real;  //массив с координатами фигуры
begin
  DelOs:=0;
  if OpenDialog1.Execute
  then
begin
   //проверка на существование файла с веденным именем
if not FileExists(OpenDialog1.FileName)
then
   begin
     MessageBox(Handle, 'Файл с заданным именем не найден! Действие отменено.',
     'Внимание!', MB_OK + MB_ICONWARNING + MB_APPLMODAL);
     Exit;
   end
   else
   begin
     //открытие выбранного файла
     FilePath:=OpenDialog1.FileName;//сохранение пути к файлу
     //1.Открытие Excel файла
E:=CreateOleObject('Excel.Application');//Создание OLE объект:
E.Workbooks.Open(FilePath);//открытие  (содержится в корневом каталоге)
     //2.1 Сохранение данных ДО  из EXCEL в Data[i]
     n:=E.ActiveWorkBook.WorkSheets.Item[1].Cells[3,1]; //сколько всего точек в ДО
//ShowMessage('всегоуглов '+FloatToStr(n));
n:=n*2+6;
     SetLength(Data,n);//выделение памяти для динамического массива
For i:=0 to (n-1) do data[i]:=E.ActiveWorkBook.WorkSheets.Item[1].Cells[3,i+1];
//3.Закрытие Excel файла
     E.quit; //Закрытие

       //2.2 Сохранение матрицы ДО с нулевыми данными
n:=Trunc(Data[5]); //Ymax+1 ткеще 0
       m:=Trunc(Data[4]); //Xmax+1 ткеще 0
SetLength(Matr,n+1,m+1);//выделение области памяти для матрицы
For i:=0 to (n) do
       For j:=0 to (m) do
Matr[i][j]:=0;
       //сохранение данных в доп массив для работы
j:=1;
       for g:=6 to (n-1) do
       begin
         Z[0,j]:=data[g];
         //ShowMessage('коорд '+FloatToStr(g)+'>>>>>>>>>   '+FloatToStr(Z[0,j]));
```

```
        j:=j+1;
            end;
            //координаты всегда с data[6] по data[n-1]
            g1:=0;//прямых
            //заполнение уравнений прямых в фигуре
        for g:=1 to Trunc(data[0]*2) do
              if g mod 2<>0 then//еслих
        begin
        g1:=g1+1;
              if g=(n-2) then//если рассматривается последний X
        begin
                Z[g1,1]:=Z[0,2]-Z[0,g+1];//a
                Z[g1,2]:=Z[0,g]-Z[0,1];   //b
                Z[g1,3]:=Z[0,g+1]*(Z[0,1]-Z[0,g])-Z[0,g]*(Z[0,2]-Z[0,g+1]) ;      //c
        if Z[0,1]<Z[0,3] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                              else   if Z[0,1]>Z[0,3] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
                if Z[0,1]=Z[0,3] then
                  if Z[0,1]<Z[0,5] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                               else   if Z[0,1]>Z[0,5] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
        end
                else
                begin
                  Z[g1,1]:=Z[0,g+3]-Z[0,g+1];//a
                  Z[g1,2]:=Z[0,g]-Z[0,g+2] ;   //b
                  Z[g1,3]:=Z[0,g+1]*(Z[0,g+2]-Z[0,g])-Z[0,g]*(Z[0,g+3]-Z[0,g+1]) ;      //c
        if Z[0,g]<Z[0,g+2] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                               else  if Z[0,g]>Z[0,g+2] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
                if Z[0,g]=Z[0,g+2] then
                  if Z[0,g]<Z[0,g+4] then Z[g1,0]:=1 //знак (в какую сторону вектор)
                                 else  if Z[0,g]>Z[0,g+4] then Z[g1,0]:=-1; //знак (в какую сторону
вектор)
        end;
                               //ShowMessage('прямая-'+FloatToStr(g1)+'
a='+FloatToStr(Z[g1,1])+' b='+FloatToStr(Z[g1,2])+' c='+FloatToStr(Z[g1,3]));
        end;
              //поиск минимума максимума новых координат фигуры
        Xmin:=0;
              Xmax:=data[4];
              Ymin:=0;
              Ymax:=data[5];
        //поиск матрицы фигуры по найденным координатам
        V:=false;
              for y2:=Trunc(Ymin) to Trunc(Ymax) do
              begin
                g:=0;//сколько 1 встроке
                for x2:=Trunc(Xmin) to Trunc(Xmax) do
        begin
                 V:=false;
                 //перебор прямых, посмотреть сколько пересечений,
```

```
                for g2:=1 to Trunc(data[0]) do  //проверка лежит ли точка на осях шаблона
        begin
                 j45:=Trunc(x2*Z[g2,1]+y2*Z[g2,2]+Z[g2,3]);
                 //ShowMessage('точка  '+FloatToStr(x2)+'  ,  '+FloatToStr(y2)+'прямая
'+FloatToStr(g2)+'>>>>>должнобыть>=0  >>>>>>>>>  '+FloatToStr(j45));
//+FloatToStr(x2)+'*'+FloatToStr(Z[g2,1])+'+'+FloatToStr(y2)+'*'+FloatToStr(Z[g2,2])+'+'+Flo
atToStr(Z[g2,3])+'='+FloatToStr(j45));
                 if  (j45=0)//and  (((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z[0,g2*2-
1])and(x2<=Z[0,g2*2+1])))  //or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-
1])and(x2>=Z[0,g2*2+1]))
                 then
                 begin
                  if
((g2=Trunc(data[0]))and((y2>=Z[0,2])and(y2<=Z[0,g2*2])and(x2>=Z[0,1])and(x2<=Z[0,g2*2-
1])))
                   then
                   begin
                    //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')  >>>>>>>
'+'конечнаяпрямая');
                    Matr[y2][x2]:=1; V:=true;  break;
                   end
                   else
                    if ((Z[g2,1]=0)and(Z[g2,3]=0)) then  //еслипрямаяпараллельнаосиX
                    begin
                     if (Z[0,g2*2-1]>Z[0,g2*2+1])and(g2<>1)then
                      if ((x2<=Z[0,g2*2-1])and(x2>=Z[0,g2*2+1]))
                      then
                      begin
                       //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')  >>>>>>>
'+'прямаяПАРАЛЛперевернутая');
                        Matr[y2][x2]:=1; V:=true;  break;
                      end
                      else
                       if (Z[0,g2*2-1]<Z[0,g2*2+1])and(g2<>1)
                       then
                        if ((x2>=Z[0,g2*2-1])and(x2<=Z[0,g2*2+1]))
                        then
                        begin
                         //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')  >>>>>>>
'+'прямаяПАРАЛЛнормальная');
                          Matr[y2][x2]:=1; V:=true;  break;
        end;
                   end
                   else //если не паралельная и не конечная то
        if((Z[g2,1]<>0)and(Z[g2,3]<>0))and((y2>=Z[0,g2*2])and(y2<=Z[0,g2*2+2])and(x2>=Z
[0,g2*2-
1])and(x2<=Z[0,g2*2+1]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])and(x2<=Z[0,g2*2-
1])and(x2>=Z[0,g2*2+1]))
                    then
                    begin
                     //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')  >>>>>>>
'+'простопрямая');
```

```
                    Matr[y2][x2]:=1; V:=true;  break;
                  end;
              end;
            end;
            if (V=false) then
      begin
              j:=0;
              SetLength(coord,n+100);//выделение памяти для динамического массива
              //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')     >>>>>>>     '+'точка
вне парал и боковых');
              for g2:=1 to Trunc(data[0]) do  //проверка лежит ли точка на осях шаблона
      begin
               if
(((Z[g2,1]<>0)and(Z[g2,2]<>0))or((Z[g2,1]<>0)and(Z[g2,2]=0)))and(((y2>=Z[0,g2*2])and(y2<
=Z[0,g2*2+2]))or((y2<=Z[0,g2*2])and(y2>=Z[0,g2*2+2])))
               then
               begin
                if             ((g2=Trunc(data[0]))and((Z[0,2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,1])<>0))     or     ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])<>0))or ((g2<>Trunc(data[0]))and((Z[0,g2*2+2]-Z[0,g2*2])<>0)and((Z[0,g2*2-1]-
Z[0,g2*2+1])=0))
                then
                begin
                 x1:=Trunc((Z[g2,2]*y2+Z[g2,3])/(-Z[g2,1]));
                 //ShowMessage('(T.      перес)x1='+FloatToStr(x1)+'                    (Расм.
T.)x2='+FloatToStr(x2)+'   >>>>>>> if x1<=x2 then j:=j+1;');
                 if x1<=x2 then
                 begin
                  if j=0 then
      begin
                   j:=j+1;//сколько точек пересечения до прямой x=0
      coord[j]:=x1;
                 end
                 else
                 begin
                  cr1:=true;
                  for cr:=1 to j do
                   if coord[cr]=x1 then begin cr1:=false; break; end;
      if cr1=true then
                  begin
                   j:=j+1;//сколько точек пересечения до прямой x=0
      coord[j]:=x1;
                  end;
                 end;
      //j:=j+1;//сколько точек пересечения до прямой x=0
      end;
                end;
               end;
              end;
            end;
            if j mod 2<>0
      then Matr[y2][x2]:=1  //если не четное
            else Matr[y2][x2]:=0;  //если четное
```

```
                //ShowMessage('пересекает до точки x='+FloatToStr(x2)+'    '+FloatToStr(j)+'
раз');
        end;
                //ShowMessage('('+FloatToStr(x2)+','+FloatToStr(y2)+')                    >>>>>>>
'+FloatToStr(Matr[y2][x2]));
              end;//x
            end;//y
          DelOs:=1; // 0;//    DelOs+
        end;
      end;
      Frame31.PageControl1.ActivePageIndex:=0;
      if DelOs=1 then
      begin
        //рисование   в КОМПАС
        // Создать объект автоматизации KOMPAS_Graphic
      if Kompas = nil then
        begin
          {$IFDEF __LIGHT_VERSION__}
          Kompas:= KompasObject( CreateOleObject('KompasLT.Application.5') );
          {$ELSE}
          Kompas:= KompasObject( CreateOleObject('Kompas.Application.5') );
          {$ENDIF}
          if Kompas <> nil then
            Kompas.Visible := false;    //невиднопокомпастконаненужна
        end;
        ForceDirectories(ProgDIR                                                    +'\show
projects'+'\'+Frame21.edit1.text+'\'+'Raw(Сырье)');//созданиепапкисименем
      //Получаем и заполняем интерфейс параметров документа
      DocumentParam:=ksDocumentParam(kompas.GetParamStruct(ko_DocumentParam));
        DocumentParam.Init();
        DocumentParam.type_:=ksDocumentFragment;//Drawing;
        //Получаеминтерфейсдокумента
        Doc2:=ksDocument2D(kompas.Document2D);
        //Создаемновыйчертеж
        Doc2.ksCreateDocument(DocumentParam);
        //создаетсяШАПКАтаблицыотображенияшаблонов
        Frame31.StringGrid3.Cells[0, 0]:='№';
        Frame31.StringGrid3.Cells[1, 0]:='Кол-во';
        Frame31.StringGrid3.Cells[2, 0]:='S фигуры';
        Frame31.StringGrid3.Cells[0, DelOs]:=IntToStr(DelOs);
        Frame31.StringGrid3.Cells[1, DelOs]:=FloatToStr(Data[1]);
        Frame31.StringGrid3.Cells[2, DelOs]:=FloatToStr(Data[2]);

      Doc2.ksText(0,Trunc(Data[5]+20),0,5,0,0,'Деловой остаток (ДО)');
      Doc2.ksText(0,(Data[5]+15),0,2,0,0,'S = '+FloatToStr(Trunc(Data[2]))+' (мм2), где S-
площадьДО;');
        Doc2.ksText(0,(Data[5]+12),0,2,0,0,'k = '+FloatToStr(Trunc(Data[1]))+' (шт.), где k-
имеющеесяколличествоДО;');
        Doc2.ksText(0,(Data[5]+9),0,2,0,0,'Xmax = '+FloatToStr(Data[4])+'; Ymax =
'+FloatToStr(Data[5]));

      //добавление картинки из папки в таблицу для отображения как в галлереи
```

```
Frame31.Image3.Canvas.Brush.Color:=clWhite;
   Frame31.Image3.Canvas.pen.Color:=clWhite;
   Frame31.Image3.Canvas.rectangle(0,0,500,500);
   Frame31.Image3.Canvas.pen.Color:=clBlack;//clSilver;//фонДО
   //Frame31.Image3.Canvas.Brush.Color:=clBlack;//clSilver;//фонДО
   Frame31.Image3.Canvas.MoveTo(0,0);     //?? x
   Frame31.Image3.Canvas.LineTo(500,0);
   Frame31.Image3.Canvas.MoveTo(0,0);  //?? y
   Frame31.Image3.Canvas.LineTo(0,500);
//Сохранение координат в массив для рисования полигона!!!!!!!!!
j:=0;
   if Data[4]>Data[5] then n:=Trunc(500/Data[4]) else n:=Trunc(500/Data[5]);
   For i:=6 to (Trunc(Data[0])*2+5) do//переборкоординат
if i mod 2=0 then
     begin
      x:=Trunc(Data[i])*n+25;//том на 100 убрать тут чтоб видно было тк размер
примера мал
Y:=Trunc(Data[i+1])*n+25;
      Frame31.Image3.Canvas.MoveTo(x,0);  //?? y
      Frame31.Image3.Canvas.LineTo(x,5);
      Frame31.Image3.Canvas.MoveTo(0,y);  //?? y
      Frame31.Image3.Canvas.LineTo(5,y);
      PointArray[j]:=Point(x,y); j:=j+1;
      //рисованиевкомпас
      if      i<(Trunc(Data[0])*2+5-1)   then   Doc2.ksLineSeg(      Trunc(Data[i]),
Trunc(Data[i+1]), Trunc(Data[i+2]),  Trunc(Data[i+3]), 1 );
      if      i=(Trunc(Data[0])*2+5-1)   then   Doc2.ksLineSeg(      Trunc(Data[i]),
Trunc(Data[i+1]), Trunc(Data[6]),  Trunc(Data[7]), 1 );
     end;
   Frame31.Image3.Canvas.Brush.Style:=bsSolid;
   Frame31.Image3.Canvas.Brush.Color:=$00E4E4E4;//$00F5F7F9;
   Frame31.Image3.Canvas.Polygon(Slice(PointArray,j));
   Frame31.Image3.Picture.SaveToFile(ProgDIR                              +'\show
projects'+'\'+Frame21.edit1.text+'\'+'Raw(Сырье)\1.bmp');
   doc2.ksSaveDocument(ProgDIR                                            +'\show
projects'+'\'+Frame21.edit1.text+'\'+'Raw(Сырье)\1.cdw');
   //закрытиефрагмента
   if doc2<>nil then doc2:=nil;
   Frame31.Image7.Picture.LoadFromFile(ProgDIR                            +'\show
projects\'+Frame21.edit1.text+'\'+'Raw(Сырье)\1.bmp');
   Frame31.Image7.Proportional:=true;
   Frame31.Image7.Transparent:=False;
   n:=0;
   //таблица
   SpeedButton1.Enabled:=false;
//добавление ДО блокируется,
   SpeedButton1.Enabled:=false;
SpeedButton2.Enabled:=true;//можнодобавлятьШаблоны
   Otch1:=true;
   Frame31.Visible:=true;//открываетсяфрейм 3
   Frame31.PageControl1.ActivePageIndex:=0;
   Frame31.PageControl1.Visible:=true;//открываетсяфрейм 3
```

```
    // Frame31.StaticText1.Visible:=False;
    //Frame31.StaticText1.Align:=alClient;
    //Frame31.StaticText1.Alignment:=taCenter;
    Frame31.StaticText2.Visible:=True;
    Frame31.StaticText2.Align:=alClient;
    Frame31.StaticText2.Alignment:=taCenter;
    Frame31.ShellListView2.Visible:=False;
    Frame31.StaticText3.Visible:=True;
    Frame31.StaticText3.Align:=alClient;
    Frame31.StaticText3.Alignment:=taCenter;
    Frame31.ShellListView4.Visible:=False;

    Frame31.StaticText4.Visible:=True;
    Frame31.StaticText4.Align:=alClient;
    Frame31.StaticText4.Alignment:=taCenter;
Frame31.ShellListView5.Visible:=False;

Frame31.StaticText5.Visible:=True;
    Frame31.StaticText5.Align:=alClient;
    Frame31.StaticText5.Alignment:=taCenter;
    Frame31.ShellListView3.Visible:=False;
//Закрытиекомпасвобщем
    if Kompas <> nil then
    begin
      //принудительнозакрыть
      Kompas.Quit;
      Kompas := nil;
    end;
end;
end;
```

ПРИЛОЖЕНИЕ Р

Листинг процедуры добавления ГО

Р.1 Добавление ГО, необходимого для раскроя

```
procedure TForm1.SpeedButton2Click(Sender: TObject);
var
  j:integer;
  RasterFormatParam:ksRasterFormatParam;
begin
  n:=Shabl;
  n:=n+1;
  razr:=false;
  if ((Shabl>=0) and (Shabl<5)) then
  begin
    Frame31.Visible:=false; //видимостьфрайм 3
frame31.Enabled:=false;//запрет на действия в листах
    Panel1.Enabled:=false;//запрет на действия в листах
    frame41.Visible:=true;//видимость панели ввода шаблонов
n0:=0;
    i:=1;
    frame41.ComboBox1.Text:='';
    frame41.LabeledEdit3.Text:='';
    frame41.LabeledEdit3.Enabled:=false;
    frame41.Button2.Enabled:=false;
    frame41.LabeledEdit1.Visible:=false;
    frame41.LabeledEdit2.Visible:=false;
    frame41.BitBtn6.Visible:=false;
    frame41.Frame61.Visible:=false;
    frame41.Frame71.Visible:=false;
    frame41.Frame81.Visible:=false;
    frame41.Frame91.Visible:=false;
    frame41.StaticText1.Visible:=false;
    frame41.StaticText2.Visible:=false;
    frame41.StaticText3.Visible:=false;
  end
  else
  begin
ShowMessage('Введено максимальное колличество шаблонов');
SpeedButton2.Enabled:=false;    //раскройзапрещен
  end;
end;
```

Р.2 Листинг кнопки «ОК» при сохранении координат (X;Y) в массив F

```
procedure TForm1.Frame41BitBtn6Click(Sender: TObject);
var
  x,y,a,b:real;
begin
  //сохранениетекущейточки
  x:=StrToFloat(Frame41.LabeledEdit1.Text);
  y:=StrToFloat(Frame41.LabeledEdit2.Text);
```

```
        //если это первая точка, то от нее начинаем рисовать линии
        n0:=n0+1; //подсчетуглов
          if n0=1 then
          begin
            a:=x;
            b:=y;
            //обнулениеэдиов+курсор
            Frame41.LabeledEdit1.Text:='';
            Frame41.LabeledEdit2.Text:='';
            Frame41.LabeledEdit1.SetFocus;
          end
          Else   //началопостроениефигуры
          begin                    //Frame41.Image1.Canvas.LineTo(Round(x+10),Round(y+10));
//далеепостроениефигуры
        //SetLength(F,n,i);//выделение области памяти для матрицы
            F[n][i]:=x;
            //SetLength(F,n,i+1);//выделение области памяти для матрицы
            F[n][i+1]:=y;
            //обнуление эдиов+курсор
        Frame41.LabeledEdit1.Text:='';
        Frame41.LabeledEdit2.Text:='';
            Frame41.LabeledEdit1.SetFocus;
        i:=i+2;
        end;
          //если больше 4 точек то можно дальше делать
        if (n0>=4) then Frame41.LabeledEdit3.Enabled:=true;
        end;
```

Р.3 Листинг процедуры выбора типа ГО из combobox

```
        procedure TForm1.Frame41ComboBox1Change(Sender: TObject);
        begin
          Frame41.LabeledEdit3.Text:='';
          Frame41.Button2.Enabled:=false;
          if Frame41.ComboBox1.Text='Произвольный' then
          begin
            razr:=false;
            n0:=0;
            frame41.LabeledEdit3.Enabled:=false;
            frame41.LabeledEdit1.Visible:=true;
            frame41.LabeledEdit2.Visible:=true;
            frame41.BitBtn6.Visible:=true;
            frame41.Frame61.Visible:=false;
            frame41.Frame71.Visible:=false;
            frame41.Frame81.Visible:=false;
            frame41.Frame91.Visible:=false;
            frame41.StaticText1.Visible:=true;
            frame41.StaticText2.Visible:=true;
            frame41.StaticText3.Visible:=true;
            Frame41.LabeledEdit1.Text:='';
            Frame41.LabeledEdit2.Text:='';
            Frame41.LabeledEdit1.SetFocus; //установкакурсора
```

```
  Frame41.LabeledEdit3.Text:='';
  Frame41.LabeledEdit3.Enabled:=false;
  Frame41.Button2.Enabled:=false;
end;
if Frame41.ComboBox1.Text='Треугольник' then
begin
  razr:=false;
  n0:=0;
  frame41.LabeledEdit3.Enabled:=true;
  frame41.LabeledEdit1.Visible:=false;
  frame41.LabeledEdit2.Visible:=false;
  frame41.Frame81.Visible:=false;
  frame41.BitBtn6.Visible:=false;
  frame41.Frame61.Visible:=true;
  frame41.Frame71.Visible:=false;
  frame41.Frame91.Visible:=false;
  frame41.StaticText1.Visible:=false;
  frame41.StaticText2.Visible:=false;
  frame41.StaticText3.Visible:=false;
  frame41.Frame61.Edit1.Text:='';
  frame41.Frame61.Edit2.Text:='';
  frame41.Frame61.Edit3.Text:='';
  frame41.Frame61.Edit1.SetFocus;
end;
if Frame41.ComboBox1.Text='Прямоугольник' then
begin
  razr:=false;
  n0:=0;
  frame41.LabeledEdit3.Enabled:=true;
  frame41.LabeledEdit1.Visible:=false;
  frame41.LabeledEdit2.Visible:=false;
  frame41.BitBtn6.Visible:=false;
  frame41.Frame71.Visible:=true;
  frame41.Frame61.Visible:=false;
  frame41.Frame81.Visible:=false;
  frame41.StaticText1.Visible:=false;
  frame41.StaticText2.Visible:=false;
  frame41.StaticText3.Visible:=false;
  frame41.Frame71.Edit1.Text:='';
  frame41.Frame71.Edit3.Text:='';
  frame41.Frame71.Edit1.SetFocus;
end;
if Frame41.ComboBox1.Text='Ромб' then
begin
  razr:=false;
  n0:=0;
  frame41.LabeledEdit3.Enabled:=true;
  frame41.LabeledEdit1.Visible:=false;
  frame41.LabeledEdit2.Visible:=false;
  frame41.BitBtn6.Visible:=false;
  frame41.Frame71.Visible:=false;
  frame41.Frame61.Visible:=false;
```

```
        frame41.Frame81.Visible:=true;
        frame41.Frame91.Visible:=false;
        frame41.StaticText1.Visible:=false;
        frame41.StaticText2.Visible:=false;
        frame41.StaticText3.Visible:=false;
        frame41.Frame81.Edit3.Text:='';
        frame41.Frame81.Edit1.Text:='';
        frame41.Frame81.Edit3.SetFocus;
      end;
      if Frame41.ComboBox1.Text='Трапеция' then
      begin
        frame41.LabeledEdit3.Enabled:=true;
        frame41.LabeledEdit1.Visible:=false;
        frame41.LabeledEdit2.Visible:=false;
        frame41.BitBtn6.Visible:=false;
        frame41.Frame71.Visible:=false;
        frame41.Frame61.Visible:=false;
        frame41.Frame81.Visible:=false;
        frame41.Frame91.Visible:=true;
        frame41.StaticText1.Visible:=false;
        frame41.StaticText2.Visible:=false;
        frame41.StaticText3.Visible:=false;
        frame41.Frame91.Edit3.Text:='';
        frame41.Frame91.Edit1.Text:='';
        frame41.Frame91.Edit2.Text:='';
        frame41.Frame91.Edit4.Text:='';
        frame41.Frame91.Edit1.SetFocus;
      end;
    end;
```

Р.4 Листинг процедур защиты системы от пользователя

```
    procedure TForm1.Frame61Edit1KeyPress(Sender: TObject; var Key: Char);
    begin
      if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
      if (Frame41.Frame61.Edit1.Text<>'')and(key=#13) then
Frame41.Frame61.Edit2.SetFocus; //Еслинажатаклавиша
Enter//Топереместитькурсорвовтороеполе
    end;

    procedure TForm1.Frame61Edit2KeyPress(Sender: TObject; var Key: Char);
    begin
      if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
      if (Frame41.Frame61.Edit2.Text<>'')and(key=#13) then
Frame41.Frame61.Edit3.SetFocus; //Еслинажатаклавиша
Enter//Топереместитькурсорвовтороеполе
    end;

    procedure TForm1.Frame61Edit3KeyPress(Sender: TObject; var Key: Char);
    begin
      if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
```

```
  if (Frame41.Frame61.Edit3.Text<>'')and(key=#13) then
Frame41.LabeledEdit3.SetFocus; //Топереместитькурсорвовтороеполе
 end;

 procedure TForm1.Frame71Edit1KeyPress(Sender: TObject; var Key: Char);
 begin
  if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
  if (Frame41.Frame71.Edit1.Text<>'')and(key=#13) then
Frame41.Frame71.Edit3.SetFocus; //Топереместитькурсорвовтороеполе
 end;

 procedure TForm1.Frame71Edit3KeyPress(Sender: TObject; var Key: Char);
 begin
  if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
  if (Frame41.Frame71.Edit3.Text<>'')and(key=#13) then
Frame41.LabeledEdit3.SetFocus; //Топереместитькурсорвовтороеполе
 end;

 procedure TForm1.Frame81Edit3KeyPress(Sender: TObject; var Key: Char);
 begin
  if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
  if (Frame41.Frame81.Edit3.Text<>'')and(key=#13) then
Frame41.Frame81.Edit1.SetFocus; //Топереместитькурсорвовтороеполе
 end;

 procedure TForm1.Frame81Edit1KeyPress(Sender: TObject; var Key: Char);
 begin
  if not (key in [#13, #8,'0'..'9']) then key:=#0;
  if (Frame41.Frame81.Edit1.Text<>'')and(key=#13) then
Frame41.LabeledEdit3.SetFocus; //Топереместитькурсорвовтороеполе
 end;

 procedure TForm1.Frame91Edit1KeyPress(Sender: TObject; var Key: Char);
 begin
  if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
  if (Frame41.Frame91.Edit1.Text<>'')and(key=#13) then
Frame41.Frame91.Edit2.SetFocus; //Топереместитькурсорвовтороеполе
 end;

 procedure TForm1.Frame91Edit2KeyPress(Sender: TObject; var Key: Char);
 begin
  if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
  if (Frame41.Frame91.Edit2.Text<>'')and(key=#13) then
Frame41.Frame91.Edit3.SetFocus; //Топереместитькурсорвовтороеполе
 end;

 procedure TForm1.Frame91Edit3KeyPress(Sender: TObject; var Key: Char);
 begin
  if not (key in [#13, #8,'0'..'9',',']) then key:=#0;
  if (Frame41.Frame91.Edit3.Text<>'')and(key=#13) //Еслинажатаклавиша Enter
 then Frame41.Frame91.Edit4.SetFocus; //То переместить курсор во второе поле
 end;
```

```
procedure TForm1.Frame91Edit4KeyPress(Sender: TObject; var Key: Char);
begin
  if not (key in [#13, #8,'0'..'9']) then key:=#0;
  if (Frame41.Frame91.Edit4.Text<>'')and(key=#13) //Еслинажатаклавиша Enter
then Frame41.LabeledEdit3.SetFocus; //То переместить курсор во второе поле
end;
```

Р.5 Листинг кнопки «далее»

```
procedure TForm1.Frame41Button2Click(Sender: TObject);
var
  a,b,c,h,x,h0,a1,b1,c1:real;
  x1,y1,m:Integer;
  Xmin,Xmax,Ymin,Ymax:real;
  imin,imax,j:integer;
begin
  Frame31.PageControl1.ActivePageIndex:=1;
  if Frame41.ComboBox1.Text='Произвольный' then
begin
    F[n][0]:=n0-1;//сохранение кол-ва точек в шаблоне
    //поиск всех необходимых данных
    //1. Поиск S фигуры    (ПОИСК ПЛОЩАДИ ФИГУРЫ  (по треугольникам с 1 до
остальных точек))
//еслитреугольник
    if n0=4 then
    begin
      a:=(sqrt(sqr(F[n][5]-F[n][1])+sqr(F[n][6]-F[n][2])));
      b:=(sqrt(sqr(F[n][3]-F[n][1])+sqr(F[n][4]-F[n][2])));
      C:=(sqrt(sqr(F[n][5]-F[n][3])+sqr(F[n][6]-F[n][4])));
      x:=(c*c+b*b-a*a)/(2*b);
      h:=(sqrt(c*c-x*x));
DataF[n][1]:=h*b/2;
    end
    else  //если многоугольник
    begin
      //поиск площади по алгоритму шнурования
a:=0;
      b:=0;
      For i:=1 to (n0)*2 do
      begin
        if (i mod 2<>0)and(i<>(n0*2-1))
        then a:=a+F[n][i]*F[n][i+3]//еслих
        else if (i mod 2=0)and(i<>(n0*2)) then b:=b+F[n][i]*F[n][i+1]; //еслиу
      end;
      DataF[n][1]:=abs(a-b)/2;//0;
end;
    //2 Поиск S прямоугольника
    Xmin:=F[n][1];
Xmax:=F[n][1];
    Ymin:=F[n][2];
Ymax:=F[n][2];
```

```
          imin:=1; //только положение Y
        imax:=2;
          For i:=3 to (n0-1)*2 do
          begin
           // Поиск Xmin,Xmax
           if i mod 2<>0 then
        begin
            if F[n][i]<=Xmin then Xmin:=F[n][i];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
        if F[n][i]>Xmax then  Xmax:=F[n][i];
           end
           // Поиск Ymin,Ymax
           else
           begin
        if F[n][i]<=Ymin then begin Ymin:=F[n][i]; imin:=i; end;//НЕ СЧИТАЕТ,
СОХРАНЯЕТ ЗН.= 0       (если начинать с 0,0 то нормально)
        if F[n][i]>Ymax then begin Ymax:=F[n][i]; imax:=i; end;
        end;
          end;
          DataF[n][2]:=Xmin;//длина прямоугольника
          DataF[n][3]:=Xmax;
          DataF[n][4]:=Ymin;//ширина прмяоугольника
        DataF[n][5]:=Ymax;
          DataF[n][6]:=(Xmax-Xmin)*(Ymax-Ymin);
          DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
          DataF[n][8]:=5;
          Shabl:=Shabl+1;
          razr:=true;
         end;
         if frame41.ComboBox1.Text='Треугольник' then
         begin
          a:=StrToFloat(frame41.Frame61.Edit1.Text);    //периметр
          b:=StrToFloat(frame41.Frame61.Edit2.Text);
          C:=StrToFloat(frame41.Frame61.Edit3.Text);
          if (a>b) and (a>c) then
          begin
           a1:=a;
           b1:=b;
           c1:=c;
          end
          else
          if (c>b) and (c>a)
          then
          begin
           a1:=c;
           b1:=b;
           c1:=a;
          end
          else
          if (b>a) and (b>c)
          then
          begin
```

```
            a1:=b;
            b1:=a;
            c1:=c;
          end
          else
          begin
            a1:=a;
        b1:=b;
            c1:=c;
          end;
          //проверка на коректность введенных данных
        //a+b>c, a+c>b, b+c>a, (a>0, b>0, c>0)
          if
(((b1+c1)>a1)and((a1>9)and(a1<501))and((b1>9)and(b1<501))and((c1>9)and(c1<501))) then
        begin
            //сохранение кординат
            F[n][0]:=3;
        F[n][1]:=0; //x1
            F[n][2]:=0; //y1
            x:=(a1*a1+b1*b1-c1*c1)/(2*a1);
            h:=sqrt(b1*b1-x*x);
            F[n][3]:=x; //x2
            F[n][4]:=h; //y2
            F[n][5]:=a1; //x3
            F[n][6]:=0; //y4
            DataF[n,1]:=a1*h/2;
            Xmin:=F[n][1];
            Xmax:=F[n][1];
            Ymin:=F[n][2];
        Ymax:=F[n][2];
            imin:=1; //только положение Y
        imax:=2;
            For i:=3 to 6 do
            begin
             // Поиск Xmin,Xmax
             if i mod 2<>0 then
        begin
              if F[n][i]<=Xmin then Xmin:=F[n][i];//НЕ СЧИТАЕТ, СОХРАНЯЕТ ЗН.= 0
(если начинать с 0,0 то нормально)
        if F[n][i]>Xmax then  Xmax:=F[n][i];
             end
             // Поиск Ymin,Ymax
             else
             begin
        if F[n][i]<=Ymin then begin Ymin:=F[n][i]; imin:=i;          end;//НЕ СЧИТАЕТ,
СОХРАНЯЕТ ЗН.= 0       (если начинать с 0,0 то нормально)
        if F[n][i]>Ymax then begin Ymax:=F[n][i]; imax:=i;    end;
        end;
            end;
            DataF[n][2]:=0;//длина прямоугольника
            DataF[n][3]:=Xmax;
            DataF[n][4]:=0;//ширина прмяоугольника
```

```
      DataF[n][5]:=Ymax;
      DataF[n][6]:=(Xmax-Xmin)*(Ymax-Ymin); {}
DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
      DataF[n][8]:=1;
      Shabl:=Shabl+1;
      razr:=true;
    end
    else
    begin
      frame41.LabeledEdit3.Text:='';
      frame41.Frame61.Edit1.Text:='';
      frame41.Frame61.Edit2.Text:='';
      frame41.Frame61.Edit3.Text:='';
      frame41.Frame61.Edit1.SetFocus;
    end;
  end;
  if frame41.ComboBox1.Text='Прямоугольник' then
  begin
    a:=StrToFloat(frame41.Frame71.Edit1.Text);    //периметр
    b:=StrToFloat(frame41.Frame71.Edit3.Text);
    if (((a<501)and(a>9))and((b<501)and(b>9))) then
begin
      //сохранение кординат
      F[n][0]:=4;
F[n][1]:=0; //x1
      F[n][2]:=0; //y1
      F[n][3]:=0; //x2
      F[n][4]:=b; //y2
      F[n][5]:=a; //x2
      F[n][6]:=b; //y2
      F[n][7]:=a; //x2
      F[n][8]:=0; //y2
      //матрицафигуры
      DataF[n,1]:=a*b; //площадь
DataF[n][2]:=0;//длина прямоугольника
      DataF[n][3]:=a;
      DataF[n][4]:=0;//ширина прмяоугольника
      DataF[n][5]:=b;
DataF[n][6]:=a*b;
      DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
      DataF[n][8]:=2;
      Shabl:=Shabl+1;
      razr:=true;
    end
    else
    begin
      frame41.Frame71.Edit1.Text:='';    //периметр
      frame41.Frame71.Edit3.Text:='';
      frame41.Frame71.Edit1.SetFocus;
    end;
  end;
  if frame41.ComboBox1.Text='Ромб' then
```

```
  begin
    a:=StrToFloat(frame41.Frame81.Edit3.Text);    //сторона
    b:=StrToFloat(frame41.Frame81.Edit1.Text);   //уголбольший
    if (((b)>90)and(b<180)and((a<300)and(a>9))) then
  begin
      //сохранение кординат
      F[n][0]:=4;
  a1:=cos((180-b)*3.14/180)*a;  //*3.14/180
      //ShowMessage(FloatToStr(a1));
      b1:=sin((180-b)*3.14/180)*a;
      //ShowMessage(FloatToStr(b1));
      F[n][1]:=0; //x1
      F[n][2]:=0; //y1
      F[n][3]:=a1; //x2
      F[n][4]:=b1; //y2
      F[n][5]:=a+a1; //x2
      F[n][6]:=b1; //y2
  F[n][7]:=a; //x2
      F[n][8]:=0; //y2
      DataF[n][2]:=0;//длина прямоугольника
  DataF[n][3]:=a+a1;
      DataF[n][4]:=0;//ширинапрмяоугольника
      DataF[n][5]:=b1;
      DataF[n][6]:=b1*(a+a1); {} //площпрям
      DataF[n,1]:=a*b1; //площадьромба
      DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
      DataF[n][8]:=3;
      Shabl:=Shabl+1;
      razr:=true;
    end
    else
    begin
      frame41.Frame81.Edit3.Text:='';
      frame41.Frame81.Edit1.Text:='';
      frame41.Frame81.Edit3.SetFocus;
    end;
  end;
  if frame41.ComboBox1.Text='Трапеция' then
  begin
    a:=StrToFloat(frame41.Frame91.Edit3.Text);
    b:=StrToFloat(frame41.Frame91.Edit1.Text);
    c:=StrToFloat(frame41.Frame91.Edit2.Text);
    c1:=StrToFloat(frame41.Frame91.Edit4.Text)*3.14/180;
    if
((c1<90)and(c1>0)and(b<a)and((a<501)and(a>9))and((b<501)and(b>9))and((c<501)and(c>9)))
then
    begin
      F[n][0]:=4;
      F[n][1]:=0; //x1
      F[n][2]:=0; //y1
      F[n][3]:=a-c*cos(c1)-b; //x2
      F[n][4]:=sin(c1)*c; //y2
```

```
      F[n][5]:=a-c*cos(c1); //x2
      F[n][6]:=sin(c1)*c; //y2
      F[n][7]:=a; //x2
F[n][8]:=0; //y2
      DataF[n][2]:=0;//длина прямоугольника
      DataF[n][3]:=a;
      DataF[n][4]:=0;//ширина прмяоугольника
DataF[n][5]:=sin(c1)*c;
      DataF[n][6]:=DataF[n][3]*DataF[n][5]; //площпрям
      DataF[n,1]:=(a+b)*(c*sin(c1))/2; //площадьтрапеции
      DataF[n][7]:=StrToFloat(frame41.LabeledEdit3.Text);
      DataF[n][8]:=4;
      Shabl:=Shabl+1;
      razr:=true;
    end
    else
    begin
      frame41.Frame91.Edit3.Text:='';
      frame41.Frame91.Edit1.Text:='';
      frame41.Frame91.Edit2.Text:='';
      frame41.Frame91.Edit4.Text:='';
      frame41.Frame91.Edit1.SetFocus;
    end;
  end;
  if (Data[2]<DataF[n,1])or(DataF[n][3]>data[4])or(DataF[n][5]>data[5]) then
  begin
    razr:=false;
    for i:=1 to 7 do DataF[n][i]:=0;
    Shabl:=Shabl-1;
    n:=n-1;
    n0:=0;
//закрытие фрейма
    frame31.Visible:=true;// действия в листах
    frame31.Enabled:=true;// действия в листах
    Panel1.Enabled:=true;//запрет на действия в листах
    frame41.Visible:=false;//видимость панели ввода шаблонов
    ShowMessage('Площадь шаблона превышает площади Делового остатка!');
end;
  if razr=true then
  begin
//рисование   в КОМПАС
    // Создать объект автоматизации KOMPAS_Graphic
if Kompas = nil then
    begin
      {$IFDEF __LIGHT_VERSION__}
      Kompas:= KompasObject( CreateOleObject('KompasLT.Application.5') );
      {$ELSE}
      Kompas:= KompasObject( CreateOleObject('Kompas.Application.5') );
      {$ENDIF}
      if Kompas <> nil then
        Kompas.Visible := false;    //невиднопокомпастконаненужна
    end;
```

```
            if n=1 then
            begin
              Frame31.StaticText2.Visible:=False;
              Frame31.ShellListView2.Visible:=true;
              ForceDirectories(ProgDIR                                                    +'\show
projects'+'\'+Frame21.edit1.text+'\'+'template(Шаблоны)');//созданиепапкисименем
              Frame31.ShellListView2.Root:=ProgDIR+'\show
projects\'+Frame21.edit1.text+'\template(Шаблоны)';
              //создаетсяШАПКАтаблицыотображенияшаблонов
              Frame31.StringGrid1.Cells[0, 0]:='№';
              Frame31.StringGrid1.Cells[1, 0]:='Кол-во';
              Frame31.StringGrid1.Cells[2, 0]:='S фигуры';
          end;
            // Создать объект автоматизации KOMPAS_Graphic  ранее
            //Получаем и заполняем интерфейс параметров документа
          DocumentParam:=ksDocumentParam(kompas.GetParamStruct(ko_DocumentParam));
            DocumentParam.Init();
            DocumentParam.type_:=ksDocumentFragment;//Drawing;
            //Получаеминтерфейсдокумента
            Doc2:=ksDocument2D(kompas.Document2D);
            //Создаемновыйчертеж
            Doc2.ksCreateDocument(DocumentParam);
            if (DataF[n,3])>(DataF[n,5]) then m:=Trunc(265/(DataF[n,3])) else
m:=Trunc(300/(DataF[n,5]));
            Frame31.Image1.Canvas.Brush.Style:=bsSolid;
            Frame31.Image1.Canvas.Brush.Color:=clWhite;//$00F5F7F9;
            Frame31.Image1.Canvas.rectangle(0,0,width,height);
            Frame31.Image1.Width:=325;//пох
            Frame31.Image1.Height:=425;//по y
            Frame31.Image1.Canvas.pen.Color:=clBlack;//$00F5F7F9;
            Frame31.Image1.Canvas.pen.Width:=1;
            Frame31.Image1.Canvas.pen.Width:=1;
            Frame31.Image1.Canvas.MoveTo(0,0);     //по x
            Frame31.Image1.Canvas.LineTo(Width,0);
            Frame31.Image1.Canvas.pen.Width:=1;
            Frame31.Image1.Canvas.MoveTo(0,0);  //по y
            Frame31.Image1.Canvas.LineTo(0,Height);
            j:=0;
            //прорисовкафигуры
            For i:=1 to Trunc(F[n,0]*2) do//(Trunc(Data[0])*2+5) do//переборкоординат
          if i mod 2<>0 then
              begin
                x1:=Trunc(F[n,i])*m+5;//том на 100 убрать тут чтоб видно было тк размер
примера мал
          Y1:=Trunc(F[n,i+1])*m+5;
                //прорисовканаосях
                Frame31.Image1.Canvas.MoveTo(x1,0);  //по y
                Frame31.Image1.Canvas.LineTo(x1,5);
                Frame31.Image1.Canvas.MoveTo(0,Y1);  //по y
                Frame31.Image1.Canvas.LineTo(5,Y1);
                PointArray[j]:=Point(x1,y1); j:=j+1;
                //рисованиевкомпас
```

```
            if  i<Trunc(F[n,0]*2-1) then Doc2.ksLineSeg(  Trunc(F[n,i]), Trunc(F[n,i+1]), Trunc(F[n,i+2]),  Trunc(F[n,i+3]), 1 );
            if  i=Trunc(F[n,0]*2-1) then Doc2.ksLineSeg(  Trunc(F[n,i]), Trunc(F[n,i+1]), Trunc(F[n,1]),  Trunc(F[n,2]), 1 );
             end;
          Frame31.Image1.Canvas.pen.Width:=3;
          Frame31.Image1.Canvas.Polygon(Slice(PointArray,j));
          //сохранениерисункавпапке Raw
          Frame31.Image1.Picture.SaveToFile(ProgDIR                    +'\show projects'+'\'+Frame21.edit1.text+'\'+'template(Шаблоны)'+'\'+IntToStr(n)+'.bmp');
       //добавление картинки из папки в таблицу для отображения как в галлереи
          //данные в компас
          Doc2.ksText(0,Trunc(DataF[n][5]+20),0,5,0,0,'Шаблон  №'+FloatToStr(n)+'  - '+frame41.ComboBox1.Text);
       Doc2.ksText(0,(DataF[n][5]+15),0,2,0,0,'S = '+FloatToStr(Trunc(DataF[n][1]))+' (мм2), где S-площадьшаблона;');
          Doc2.ksText(0,(DataF[n][5]+12),0,2,0,0,'k  =  '+FloatToStr(Trunc(DataF[n][7]))+' (шт.), где k-необходимоеколличествошаблонов;');
          Doc2.ksText(0,(DataF[n][5]+9),0,2,0,0,'Xmax = '+FloatToStr(DataF[n][3])+'; Ymax = '+FloatToStr(DataF[n][3]));
          Frame31.StringGrid1.Cells[0, n]:=IntToStr(n);
          Frame31.StringGrid1.Cells[1, n]:=FloatToStr(Trunc(DataF[n][7]));
          Frame31.StringGrid1.Cells[2, n]:=FloatToStr(Trunc(DataF[n][1]));
       //закрытие фрейма
          frame31.Enabled:=true;// действия в листах
          Panel1.Enabled:=true;//запрет на действия в листах
          frame41.Visible:=false;//видимость панели ввода шаблонов
          //записать данные в таблицу для вывода пользователю
       if  n=1  then  begin SpeedButton2.Enabled:=true; SpeedButton3.Enabled:=true; SpeedButton4.Enabled:=true; SpeedButton5.Enabled:=true;end;//раскройразрешен
          Otch2:=true;
          Frame31.PageControl1.Visible:=true;//открываетсяфрейм 3
          Frame31.Visible:=true; //видимостьфрайм 3
          //SaveAsToRasterFormat!!!!!!!!!!!!!!!!
          doc2.ksSaveDocument(ProgDIR                    +'\show projects'+'\'+Frame21.edit1.text+'\'+'template(Шаблоны)'+'\'+IntToStr(n)+'.cdw');
          //закрытиефрагмента
          if doc2<>nil then doc2:=nil;
          Frame31.Image6.Picture.LoadFromFile(ProgDIR                    +'\show projects\'+Frame21.edit1.text+'\'+'template(Шаблоны)'+'\'+IntToStr(n)+'.bmp');
          Frame31.Image6.Proportional:=true;
          Frame31.Image6.Transparent:=False;
       h11:=shabl;
          //Закрытие компас в общем
       if Kompas <> nil then
          begin
            //принудительнозакрыть
            Kompas.Quit;
            Kompas := nil;
          end;
        end;
       end;
```

Р.6 Листинг кнопки «отмена»

```
procedure TForm1.Frame41Button1Click(Sender: TObject);
begin
  for i:=1 to 7 do DataF[n][i]:=0;
  Shabl:=Shabl;
  n:=n-1;
  n0:=0;
//закрытие фрейма
  frame31.Visible:=true;// действия в листах
  frame31.Enabled:=true;// действия в листах
  Panel1.Enabled:=true;//запрет на действия в листах
  frame41.Visible:=false;//видимость панели ввода шаблонов
end;
```

ПРИЛОЖЕНИЕ С

Код кнопки «новый проект»

```
procedure TForm1.N2Click(Sender: TObject);
var
  AMsgDialog: TForm;
  ctrl : TControl;
begin
  if q=true then
  begin
    AMsgDialog := CreateMessageDialog('Название: '+Frame21.edit1.text+#13+'Автор:
'+Frame21.edit2.text+#13+'Закрытьпроект?', mtWarning, [mbYes, mbNo]);
    with AMsgDialog do
    try
      Caption := '!' ;
      Height := 140;
      Ctrl:=FindChildControl('Yes');
      Case ShowModal of
        ID_YES:
        begin
          Frame21.edit1.text:='';
          Frame21.edit2.text:='';
          //креате
          FormCreate(Sender);
          q:=false;
end;
          {кнопка сохранение}
ID_NO:                    {DeleteFolder('C:\Users\Сергей\Desktop\KR_Demo2.1\show
projects'+'\'+Frame21.edit1.text)};
        end;
      finally
        Free;
      end;
    end
    else
    begin
      Frame21.edit1.text:='';
      Frame21.edit2.text:='';
      //креате
      FormCreate(Sender);
      q:=false;
end;
end;
```

ПРИЛОЖЕНИЕ Т

Программное обеспечение «Pazl2D-v1»

Прилагается диск программы«Pazl2D-v1» с примерами созданных карт раскроя ДО.

ПРИЛОЖЕНИЕ У

Руководство пользователю

Прилагается диск с руководством пользователю в формате (.pdf).