O’ZBEKISTON RESPUBLIKASI
OLIY VA O’RTA MAXSUS TA’LIM VAZIRLIGI
MIRZO ULUG`BEK nomidagi O’ZBEKISTON MILLIY UNIVERSITETI
“MEXANIKA-MATEMATIKA” FAKULTETI
“Programmalash va tarmoq texnologiyalari” kafedrasi

мavzusidagi

BITIRUV MALAKAVIY ISH

Ilmiy rahbar: dotsent Hakimov M.X.
Bajardi: “Amaliy matematika va informatika” yo’nalishi bitiruvchisi
Fayzullayeva Zarnigor Inatillayevna

“Bitiruv malakaviy ish kafedradan dastlabki himoyadan o’tdi”
17- sonli bayonnoma 22 may 2014- yil

Toshkent – 2014

Содержание

Введение

В науке постоянно приходится сталкиваться с проблемой расчета систем, имеющих сложную геометрическую конфигурацию и нерегулярную физическую структуру. С помощью компьютера можно выполнить расчеты приближенных численных методов. Один из методов является Метод конечных элементов (МКЭ). В последние годы Метод конечных элементов занял ведущее положение и получил широкое применение. Метод конечных элементов (МКЭ) — это численный метод решения задач прикладной механики.

МКЭ широко используется для решения задач механики деформируемого твёрдого тела, тепло обмена, гидродинамики и электромагнитных полей. Идея метода конечных элементов заключается в том, что минимизация функционала вариационной задачи осуществляется на совокупности функций, каждая из которых определена на своей подобласти, для численного анализа системы позволяет рассматривать его как одну из конкретных ветвей диаоптики — общего метода исследования систем путём их расчленения. Метод конечных элементов возник с решением задач космических исследований в 1950-х годах. Этот метод возник из строительной механики и теории упругости. Существенный толчок в своём развитии МКЭ получил в 1963 году после того, как было доказано то, что его можно рассматривать, как один из вариантов распространённого в строительной механике метода Рэлея-Ритца, который путём минимизации потенциальной энергии сводит задачу к системе дифференциальных уравнений или систем дифференциальных уравнений. линейных уравнений равновесия. После того, как была установлена связь МКЭ с процедурой минимизации, он стал применяться к задачам описываемым уравнениями Лапласа или Пуассона. Область применения

МКЭ значительно расширилась, когда было установлено (в 1968 году), что уравнения, определяющие элементы в задачах могут быть легко получены с помощью вариантов метода взвешенных невязок, таких как метод Галёркина, или метод наименьших квадратов. Это сыграло важную роль в теоретическом обосновании МКЭ, так как позволило применять его при решении многих типов дифференциальных уравнений. Таким образом, метод конечных элементов превратился в общий метод численного решения

Практически все современные расчёты на прочность проводят, используя метод конечных элементов.

Инженерные конструкции можно рассматривать как некоторую совокупность конструктивных элементов, соединенных в конечном числе узловых точек. Если известны соотношения между силами и перемещениями для каждого отдельного элемента, то, используя хорошо известные приемы строительной механики, можно описать свойства и исследовать поведение конструкции в целом.

В сплошной среде число точек связи бесконечно, и именно это составляет основную трудность получения численных решений в теории упругости. Понятие конечных элементов, введенное впервые Тернером и др., представляет собой попытку преодолеть эту трудность путем разбиения сплошного тела на Отдельные элементы, взаимодействующие между собой только в узловых точках, в которых вводятся фиктивные силы, эквивалентные поверхностным напряжениям, распределенным по границам" элементов. Если такая идеализация допустима, то задача сводится к обычной задаче строительной механики, которая может быть решена численно.

На первый взгляд, этот интуитивно понятный и доступный инженерный метод выглядит не совсем убедительно в частности, остается открытым вопрос о соотношениях между силами и перемещениями отдельных элементов. На данном же этапе целесообразно кратко описать

общий метод расчета конструкций, который будет широко использоваться в книге после рассмотрения свойств конечных элементов.

В дальнейшем будет показано, что метод конечных элементов применим и ко многим задачам иного типа, но и тогда основные свойства элемента выражаются в форме, принятой в строительной механике.

§ 1. Постановка задачи

Рассматривается задача определение начального фронта для упорядочения номеров узлов дискретной модели двумерного тела сложной конфигурации.

С этой цели предлагается алгоритм определения начального фронта которой позволяет минимизировать ширину ленты разрешающей системы уравнений метода конечных элементов. Для решение задачи введем понятие начального фронта. Для этого определим понятие вершины, граничных и внутренних точек. Вершина – это точка которая встречается дискретной модели тела всего один раз. Граничные точки – это точки которые расположены на внешней границы тела. Они встречаются в дискретной модели тела два раза. Внутренние точки располагаются внутри двумерной области и встречаются в дискретной модели четыре раза.

Начальный фронт представляет собой номера узлов, которые формируется следующим образом:

вершина – граничная точка - … - граничная точка – вершина.

На начальном этапе формируется три массива данных:

M1[1…n1] – массив, состоящей из вершин, где n1 – количество вершин.

M2[1…n2] – массив, состоящей из граничных точек, где n2 – количество граничных точек.

M3[1…n3] – массив, состоящей из внутренних точек, где n3 – количество внутренних точек.

Алгоритм определения начального фронта представляет собой определение последовательности точек соответствующую выше описанному правилу. Таким образом, из множество M1 определяются возможные комбинации из номеров вершин, а граничные точки из массива M2, которые соединяют эти вершины, определяются посредством поиска в массиве MN.

Заключительным этапом решение задачи является применение фронтального лента минимизации ширины по каждому из этих начальных фронтов. Эффективная ширина ленты соответствует минимальному из них.

Дана двумерная область $\boldsymbol{\Omega}$ (Рис. 1), для которой необходимо составить дискретную модель, при этом используется метод сшивания блоков данной сложной области, которые получаются при разбиении сложной области на блоки.

Элементарной областью называется область, для которой существуют алгоритмы разбиения на конечные элементы.

Дискретизация области включает задание числа, размеров и формы подобластей ($\boldsymbol{\Omega_n}$), которые используются для построения дискретной модели реального тела. При этом, с одной стороны, элементы должны быть выбраны достаточно малыми, чтобы получить приемлемые результаты, а с другой стороны, применение достаточно крупных элементов сокращает вычислительную работу.

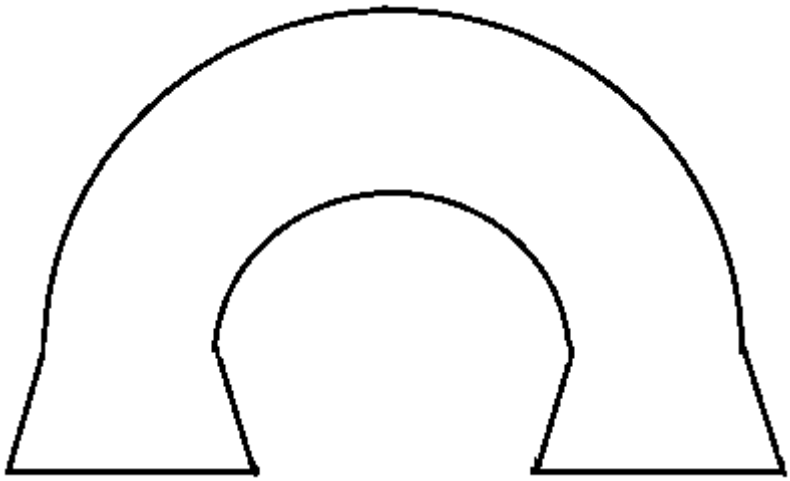


Рисунок 1. Исходная область

В общем случае, если имеется n-подобластей, то область будет просуммирована по подобластям, т.е. согласно формуле:

$$\Omega = \sum_{i=1}^{n} \Omega_i$$

Основная проблема, которая встаёт при составлении дискретной модели области **Ω** является сшивание подобластей **Ω_1, Ω_2, Ω_3, ... , Ω_n.**

1. Исходная область разбивается на несколько подобластей (Рис.2): **Ω1, Ω2, Ω3, Ω4**

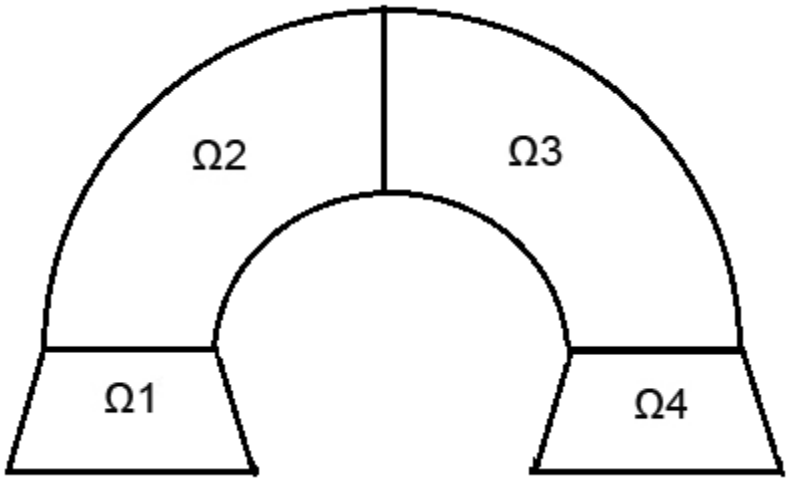


Рисунок 2. Разбиение области на элементарные подобласти

Плохое или несовершенное разбиение будет приводить к ошибочным результатам, если даже остальные этапы осуществляются с достаточной точностью. Для составления дискретной модели данной сложной области необходимо рассмотреть 2 типа подобластей и составить для них дискретные модели, а затем просто произвести поочерёдное сшивание остальных идентичных подобластей, и в итоге мы получим дискретную модель всей сложной области. Наша сложная область состоит из таких подобластей

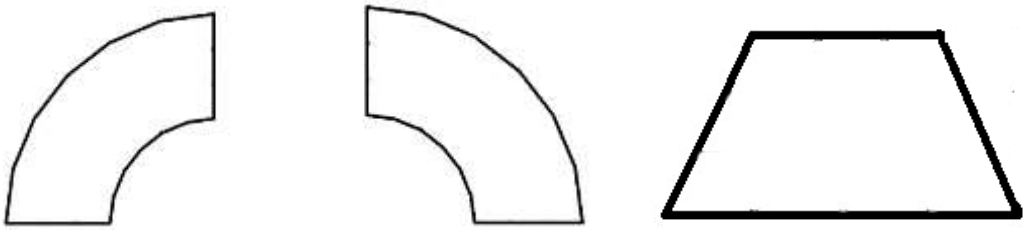


Рисунок 3. Элементарные подобласти

Алгоритмы дискретизации элементарных областей

Отделение области на конечные элементы, осуществляется разбиением на элементарные области данную область. Для каждой области подходит следующая характеристика

Ω = { N, M, MN}

где,

N – Число узлов точек;

M – Число конечных элементов;

MN –Номера узлов точек каждого конечного элемента;

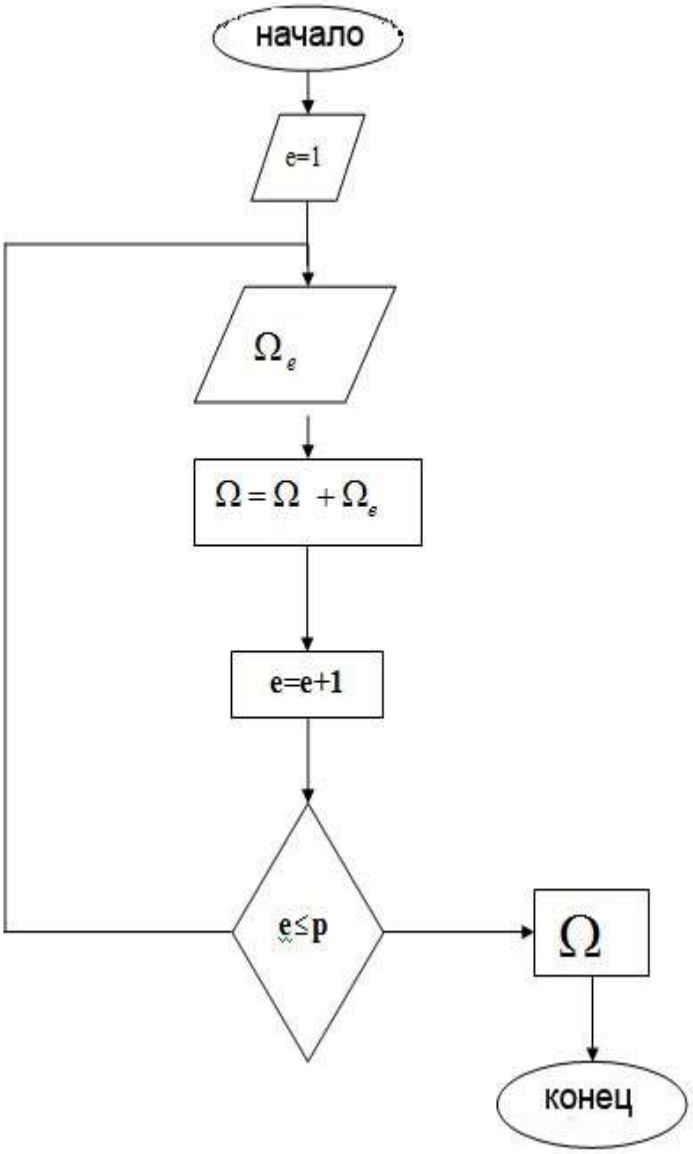
Ω -дискретная модель рассматриваемой области;

здесь,

N= (m+1)(n+1);

M= (n*m);

Алгоритм процесса дискретизации сложной области



§ 2. Алгоритм определения начального фронта

Для определения начального фронта нам требуется ввести понятия вершина, крайние и граничные точки. Начальный фронт определяется следующим образом:

<вершина> + <крайняя точка> + … + < крайняя точка > <вершина>

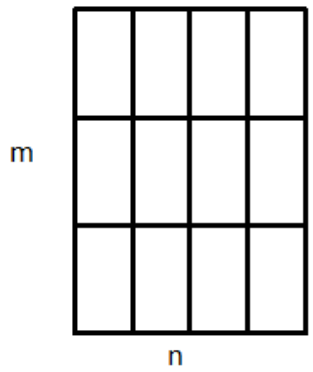
Рассмотрим начальный фронт в одном конкретном примере. Для этого вводим данные области:

m=3;

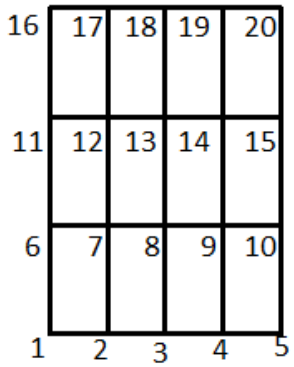
n=4;

A(0,0), B(3,0), C(0,4), D(4,3)

Результат:



Введем нумерацию:



Для нахождения начального фронта требуется найти вершину, край, и внутренние точки области.

M1 = {1, 5, 16, 20}; - вершины

M2 = {2, 3, 4, 6, 10, 11, 15, 17, 18, 19}; - крайние точки

M3 = {7, 8, 9, 12, 13, 14}; - внутренние точки

Теперь по определению находим начальный фронт:

1 » {2, 6, 7}

1 » 2 » {3, 7, 8}

1 » 2 » 3 » {4, 8, 9}

1 » 2 » 3 » 4 » {5, 9, 10}

1 » 2 » 3 » 4 » 5;

Здесь нашли первый начальный фронт

1) 1 » 2 » 3 » 4 » 5;

   Точки 1 , 2, 3, 4, 5 являются начальным фронтом.

Таким образом, находим следующей начальных фронтов.

2) 1 » 6 » 11 » 16;
3) 16 » 17 » 18 » 19 » 20;
4) 5 » 10 » 15 » 20;

В итоге мы имеем 4 начальных фронтов:

N1 = {1,2,3,4,5};

N2 = {1,6,11,16};

N3 = {5,10,15,20};

N4 = {16,17,18,19,20}

§ 3. Алгоритм дискретизации произвольной области

§ 3.1 Алгоритм дискретизации произвольной четырехугольной области

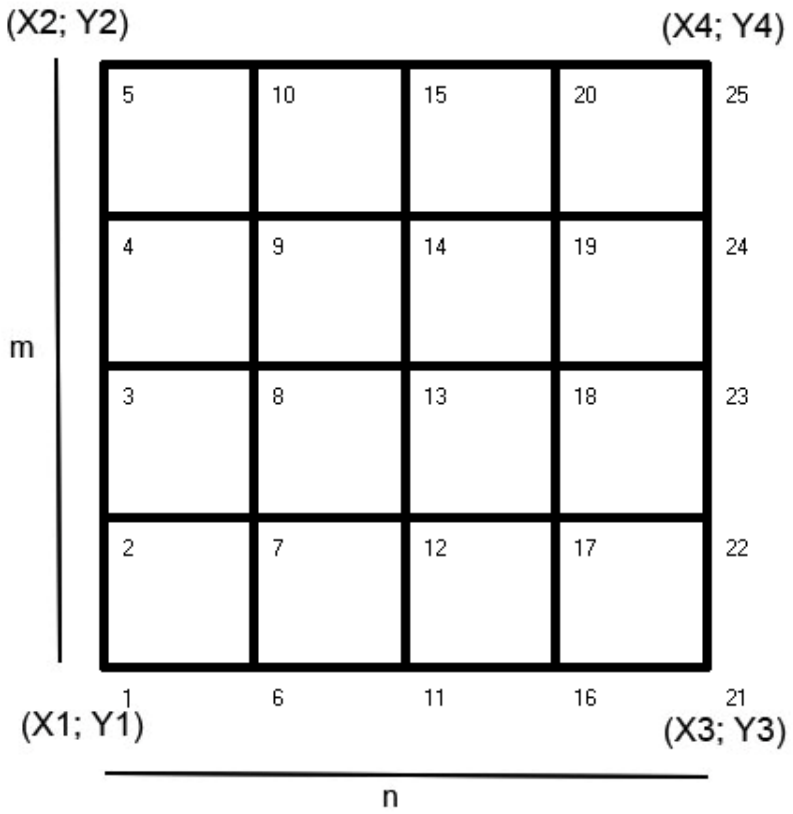


Рисунок 4. Разбиение четырехугольника на конечные элементы

Данная область(Тип 1) называется прямолинейной прямоугольной областью или областью, образованной из конечных элементов (Рис. 4). Для создания прямолинейной прямоугольной области нам понадобится следующее:

1. координаты точек прямолинейного прямоугольника; ((x_1, y_1) , (x_2, y_2) , (x_3, y_3) , (x_4, y_4)).
2. Число разбиений (m=3) сторон прямолинейного прямоугольника по оси Ox;
3. Число разбиений (n=3) сторон прямолинейного прямоугольника по оси *Oy*;

На основе вышеприведенной информации создание дискретной модели элементарной области осуществляется следующим образом

1.Отрезки (x_1, y_1) и (x_2, y_2) разделим на m равных частей. Далее применим следующую формулу

$$x = \frac{x_1 + \gamma_i x_2}{1+\gamma_i} \quad \text{и} \quad y = \frac{y_1 + \gamma_i y_2}{1+\gamma_i} \tag{1}$$

Здесь,

$$\gamma_i = \frac{i}{m-i};\ i = \overline{1, m-1}.$$

1. Координаты точек, полученные при разбиении области на конечные элементы с начальными точками заносят в массив **M1[1…m+1,1…2]**

2. Отрезки (x_3, y_3) $(x_4, y)_4$ тоже разделяем на равные части, применяя формулу (1). Координаты точек заносим в массив M2[1…m+1,1…2]

3. Затем, берем с первого по $m+1$ значения этих двух массивов, сопоставляя каждые точки по формуле (1), разделяем на n равных частей

В результате, для образования дискретной модели образованных конечных элементов, отделяем массив **MN [1…M,1…4]** . Здесь для каждого конечного элемента находятся подходящие узловые точки, в итоге:

Число узловых точек области N: **N=(3+1)(3+1)=16**

Число конечных элементов M: **M=4*3=12**

Ниже представлен алгоритм дискретизации четырехугольной области в виде блок-схемы:

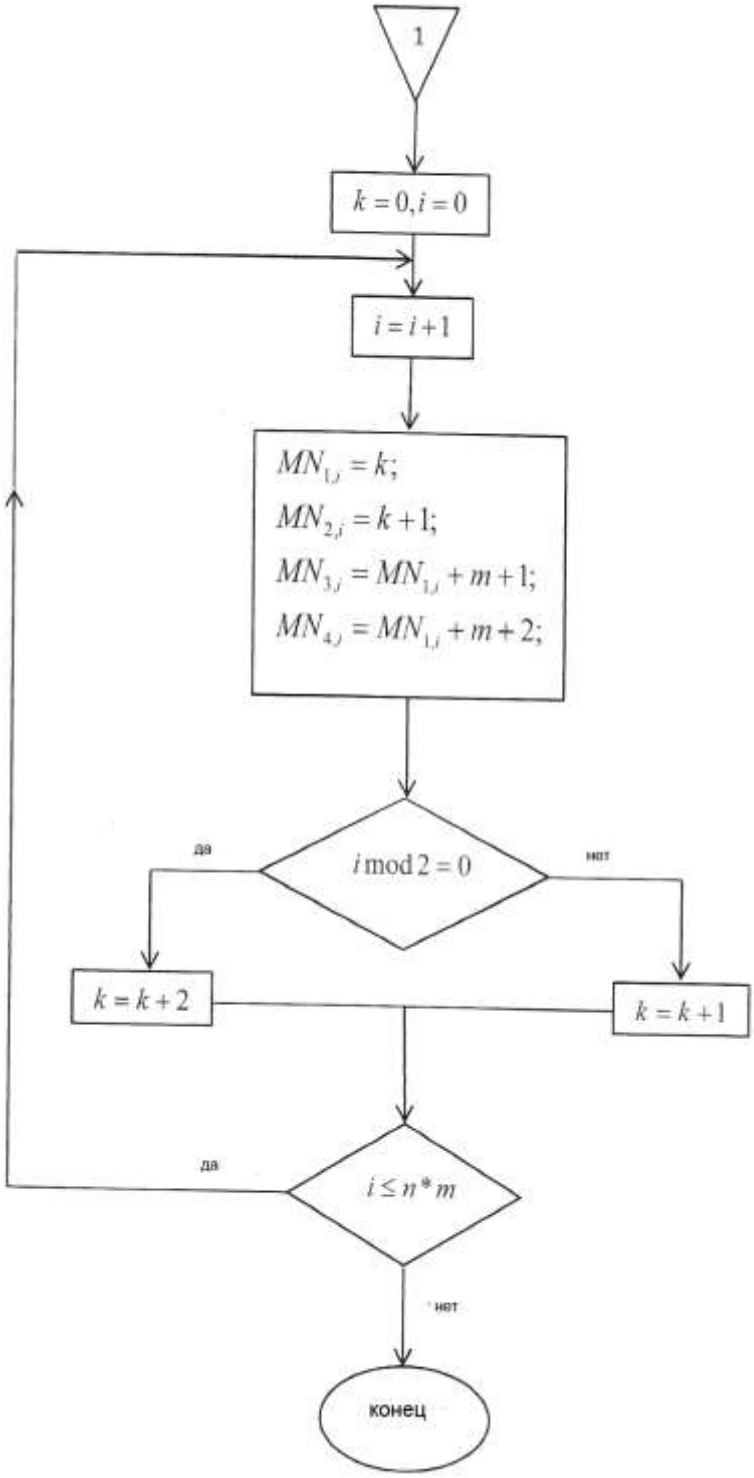


Рисунок 5. Блок-схема разбиения четырехугольника

§ 3.2 Алгоритм дискретизации криволинейной области

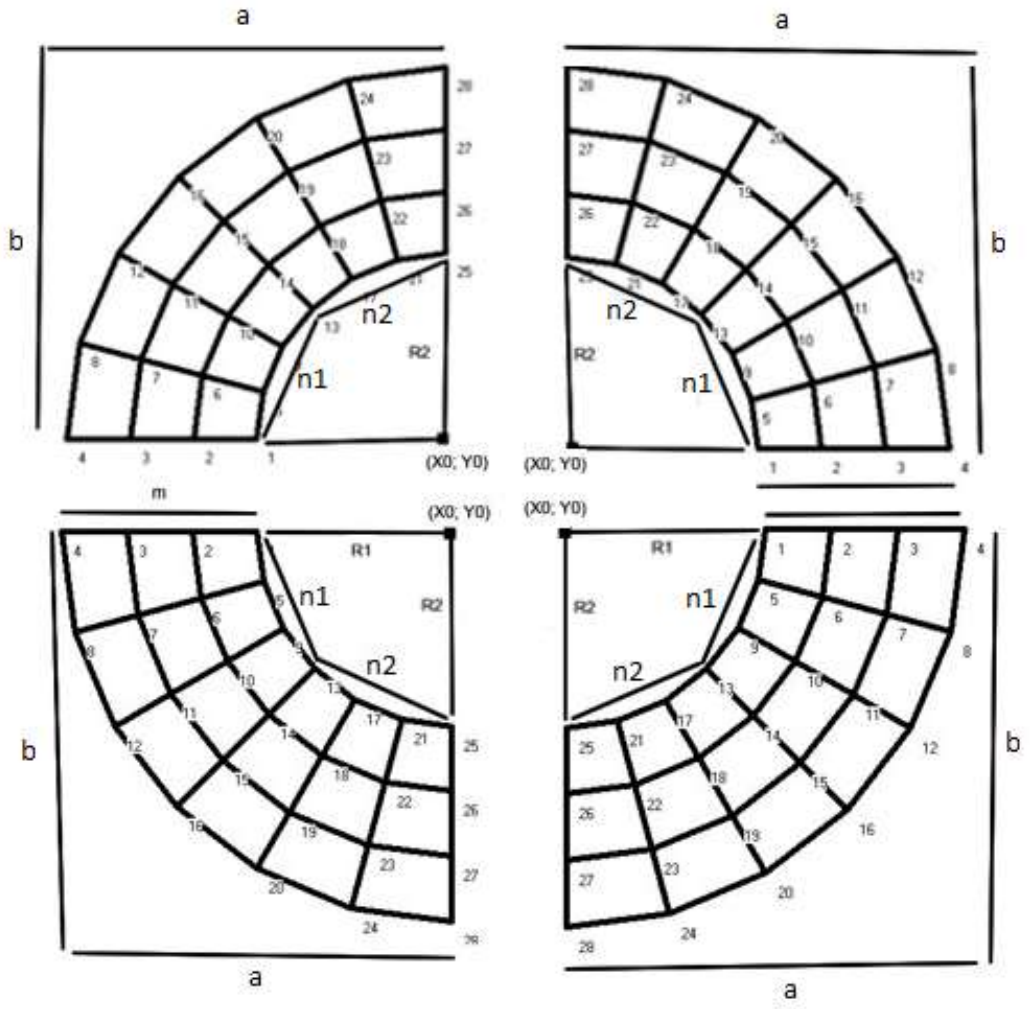


Рисунок 6. Разбиение элементарной области на конечные элементы

Здесь (Рис. 6)также сначала область (Тип 2) разделяется на квадраты, и в результате этого приводится к типу 2 , потом строится ее дискретная модель, для этого нам потребуется следующее:

1. O _ Центр координат. $O(x, y)$

2. r_1, r_2 – внутренние радиусы

3. a – высота

4. b – ширина

5. m – Число разбиений по оси Oy

6. $n1$, $n2$ – Число разбиений по оси Oy *и* Ox

7. dist_a = {a1, a2, a3, . . . , a$_m$} – массив расстояний между узловыми точками**;**

Вот ее блок-схема:

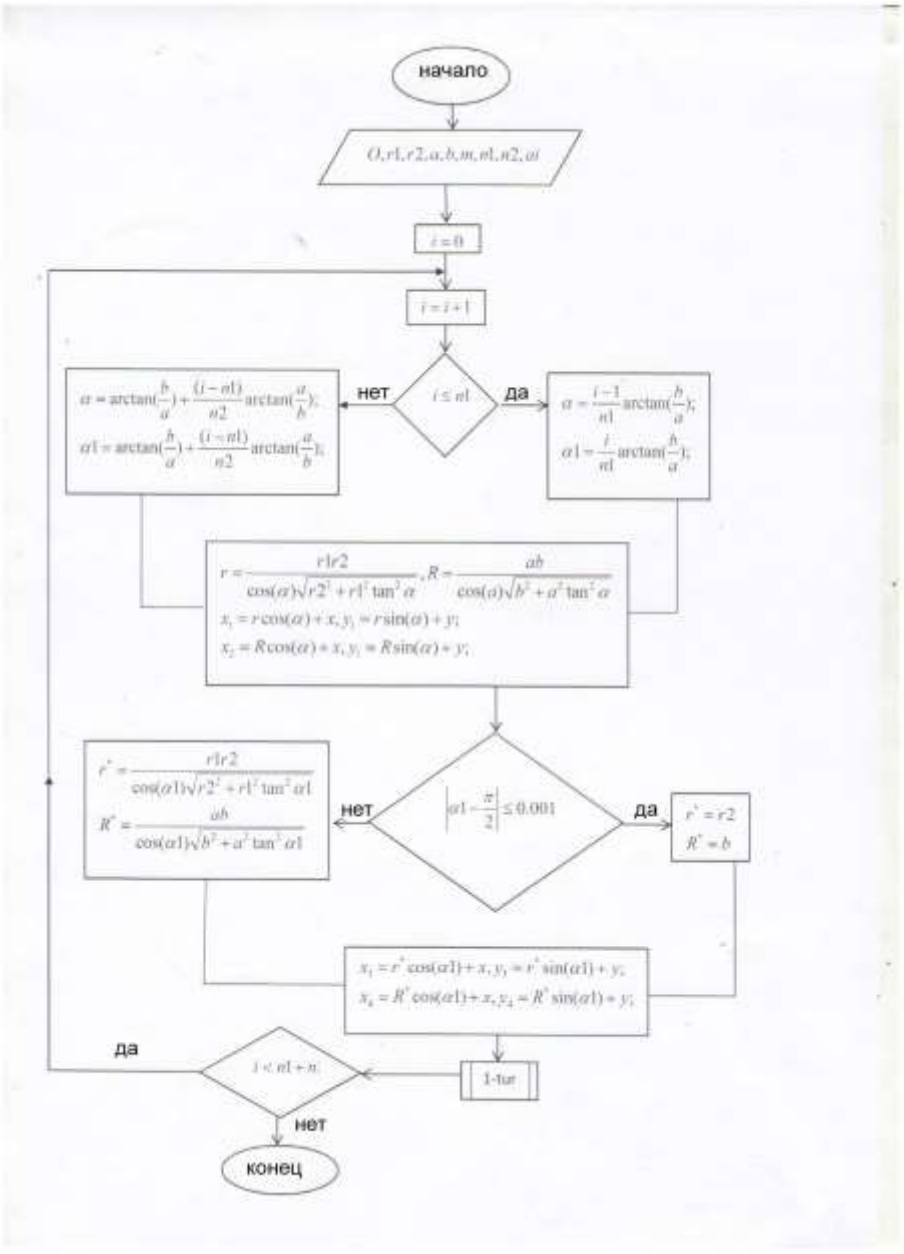


Рисунок 7. Блок-схема разбиения области

§ 4 . Алгоритм сшивания областей

После вычисления всех необходимых значений можно произвести сшивание подобластей, что предполагает перенумерацию узловых точек, перенумерацию узлов в конечных элементах, а также определения некоторых общих точек, лежащих на границах.

В результате мы получаем область

$$\Omega = \{N, M, MN\}$$

из $\Omega = \Omega_1 + \Omega_2 + \Omega_3 + \Omega_4$ или $\Omega = \sum_{i=1}^{n} \Omega_i$

При сшивании происходит копирование первой подобласти (Ω_1) в общую, а для передачи второй подобласти (Ω_2) необходимо производить перенумерацию.

1. Сперва необходимо найти общие узловые точки подобластей Ω_1 и Ω_2, т. е. узловые точки координаты (x;y) которых совпадают и записать их в массив, для дальнейшего сравнения.

В нашем примере таковыми узлами являются:

из Ω_1 {1, 2, 3, 4}

из Ω_2 {1, 2, 3, 4}

2. При сшивании номера узлов из подобласти Ω_2 изменяются на номера соответствующие совпадающие номера узлов из подобласти Ω_1 , а

перенумерация оставшихся номеров узлов из подобласти Ω_2 определяется следующей формулой:

$$N = Ni + i - q$$

Где значения :

Ni – максимальный номер узловой точки из подобласти Ω_1 (или количество узлов Ω_1);

i – номер узловой точки из подобласти Ω_2 , которую необходимо перенумеровать;

q – количество совпадающих (граничных) узловых точек из подобласти Ω_2, меньших по значению чем i.

Например новое значение номера узловой точки {5} из подобласти Ω_2 равно:

$$N = 16 + 5 - 4 = 17;$$

Для {6}

$$N = 16 + 6 - 4 = 18;$$

и т. д. пока все узловые точки подобласти Ω_2 не будут перенумерованы данным способом. В результате, после перенумерации область имеет вид

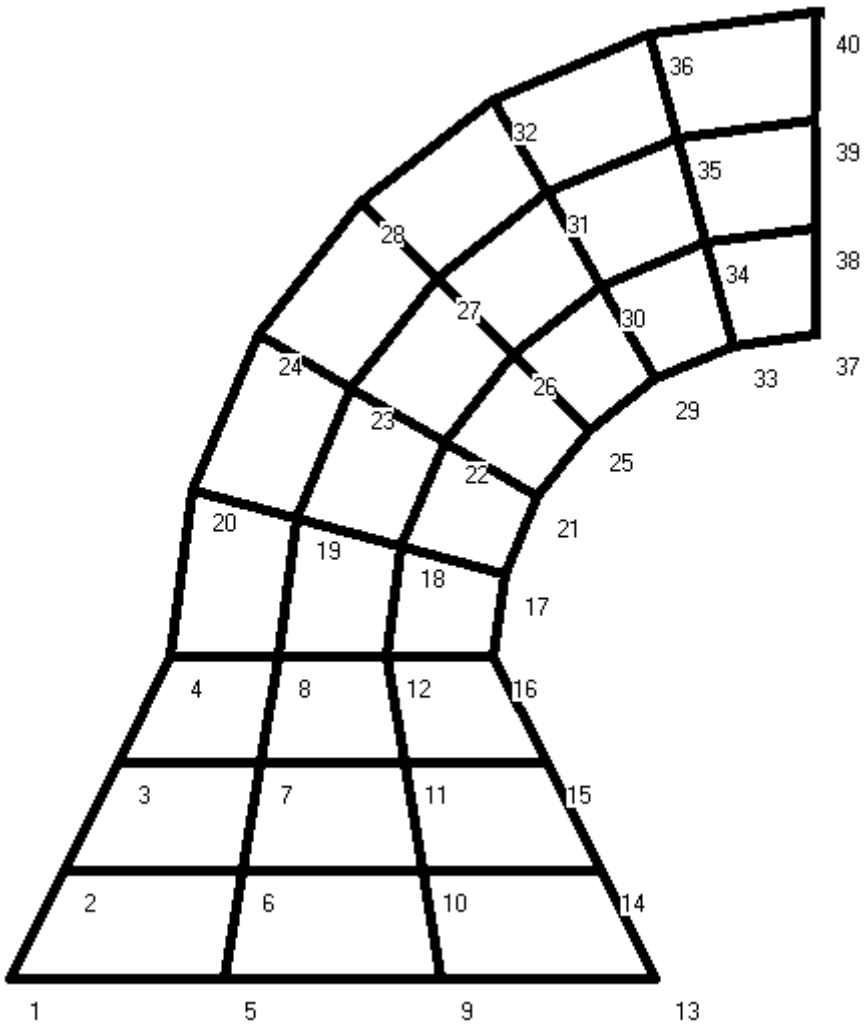


Рисунок 8 Сшивание двух подобластей

В результате получаются матрицы MK и MN области **Ω** которые представляют собой:

Элементы матрицы MK (координаты узловых точек) берутся из матриц MK1 и MK2, с учетом того что индексы координат узловых точек области **Ω** меняются на соответствующие номера узловых точек, и имеет вид:

$$MK = \begin{vmatrix} x1 & y1 \\ x2 & y2 \\ \cdots\cdots & \cdots\cdots \\ x60 & y60 \end{vmatrix}$$

Изменяется также и матрица MN (конечные элементы), часть матрицы MN полностью берётся из матрицы MN1, а остальная часть матрицы изменяется в соответствии с перенумерацией

MN

1	2	3	4
5	6	7	8
.....			
55	59	56	60

$$\mathbf{N = N1 + N2 - q}$$

Где N1 и N2 – количество узловых точек 1 и 2 подобластей, а q – количество граничных или же совпадающих узловых точек.

Значение M (количество элементов) области **Ω** равно:

$$\mathbf{M = M1 + M2}$$

Где M1 и M2 – количество элементов 1 и 2 подобластей.

Далее таким же методом объединяем области **Ω** и **Ω3,** и получаем новую область **Ω** с новыми значениями N, M, MK, MN, и т.д. с областью **Ω**8.

В итоге на выходе мы имеем значения для области **Ω** {N, M, MK, MN}

4)

MN

1	2	3	4
5	6	7	8
.....			
72	73	75	76
43	44	73	76

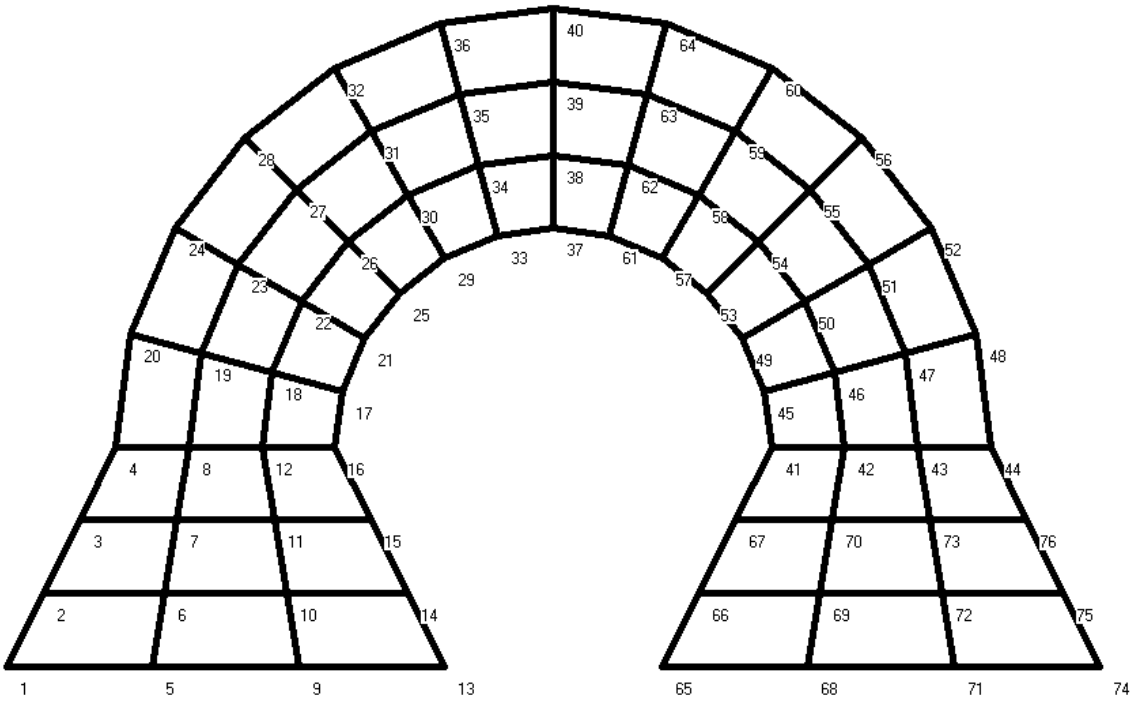


Рисунок 9. Разбиение исходной области на конечные элементы

§ 5. Описание программного обеспечения

В вышеприведенных параграфах мы расматривали построение алгоритмов элементарной области, сложной области и их дискретную модель. Пр создании дискретной модели сложной системы, разбили ее на несколько элементарных областей, а для каждой элементарной области составили дискретную модель, и с помощью этих составленных моделей образовали дискретнуюю модель сложной области.

В этом параграфе приведем коплекс программ, образующих дискретную модель сложной области. Эта комплексная программа была составлена с помошью языка программирования С++.

Возможности языка программирования С++ приведены в приложении

Таким образом кратко привели возможности программирования С++ Builder 6. теперь ниже приведем следующие понятия: интерфейс комплекса программ, образующий дискретную модель сложной области, из каких компонентов составили и как ее надо использовать. Основной интерфейс программы

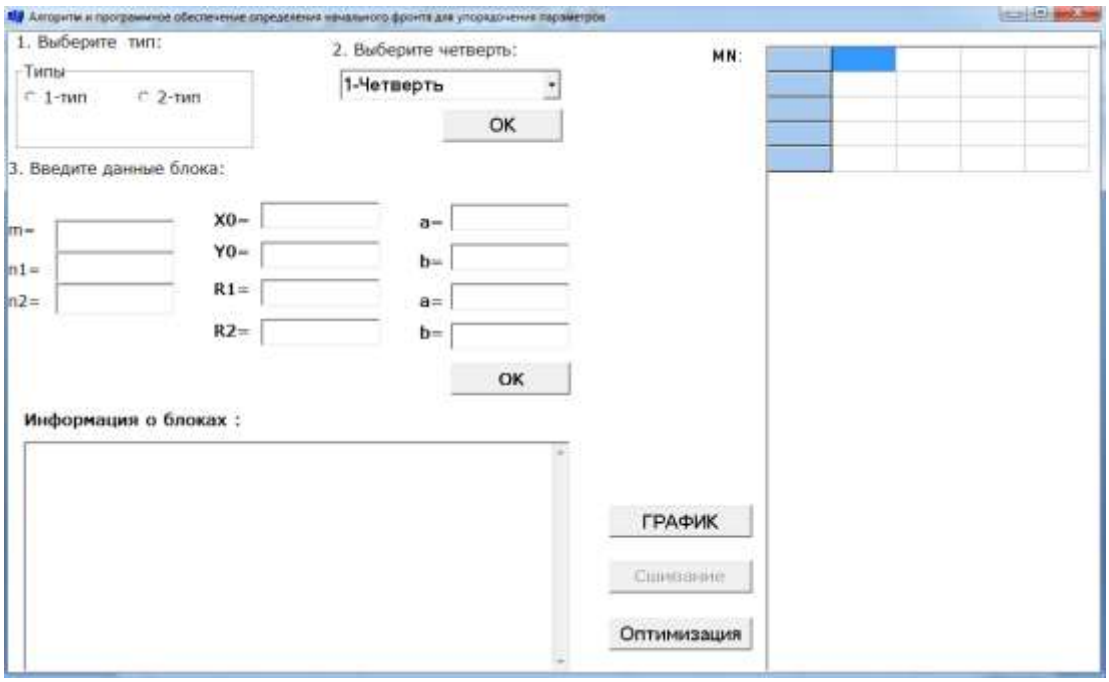


Рисунок 10. Интерфейс программного обеспечения

1. Выбор типа вводящего блока (1-произвольный квадрат, 2-полукруг)
2. Выбор четверти типа
3. Ввод числа разбиений
4. СШИВАНИЕ - соединение областей
5. Сведения по блокам.
6. MN-номер узловых точек, каждого граничного элемента.
7. График.

Рисунок 11. Представление формы

1. Выбор типа чертежа:
   - чертить все блоки.
   - чертить последний соединенный блок.
2. Вывести на экран выбранные блоки.
3. Вывести на экран с вершинами элементов.

§ 6. Контрольно - тестовый пример

Контрольный тестовый пример подразумевает собой демонстрацию поставленной задачи на ЭВМ, в наглядной и понятной форме.

В самом начале введем начальные значения для первого типа:

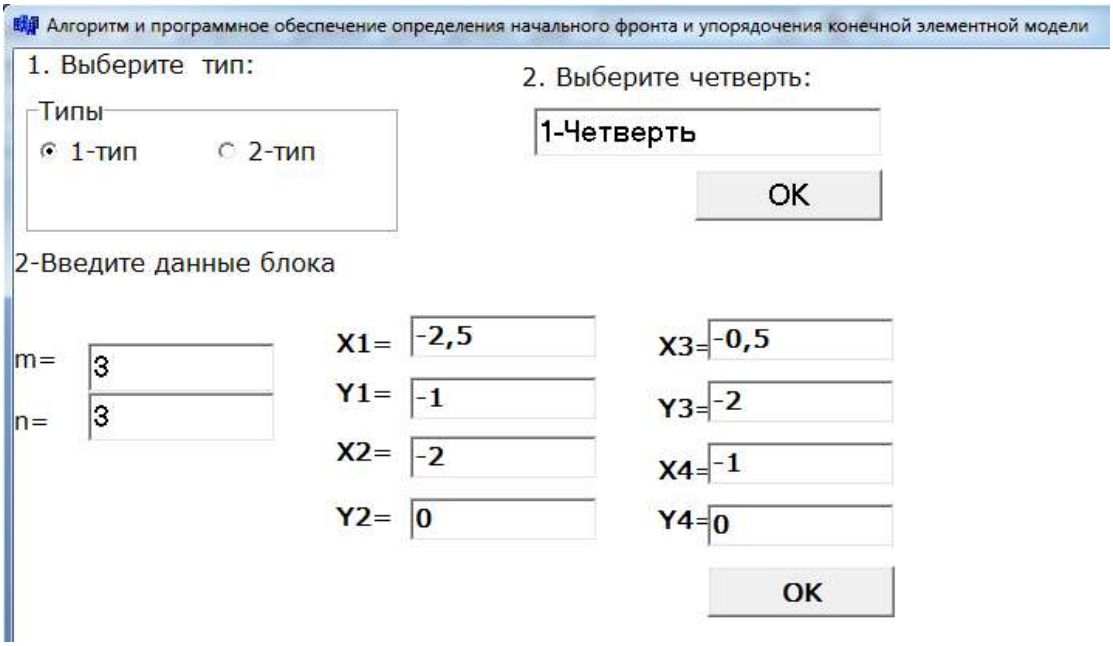


Рисунок 12. Часть интерфейса

Затем, после нажатия кнопки "ГРАФИК" мы увидим график первой подобласти :

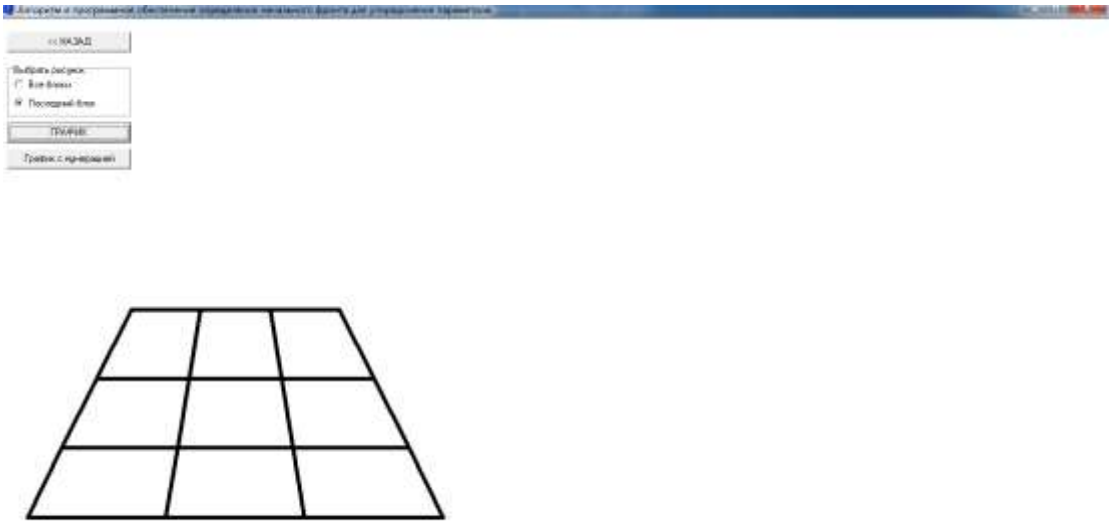


Рисунок 13. Дискретная модель

Далее мы вводим данные для второй подобласти:

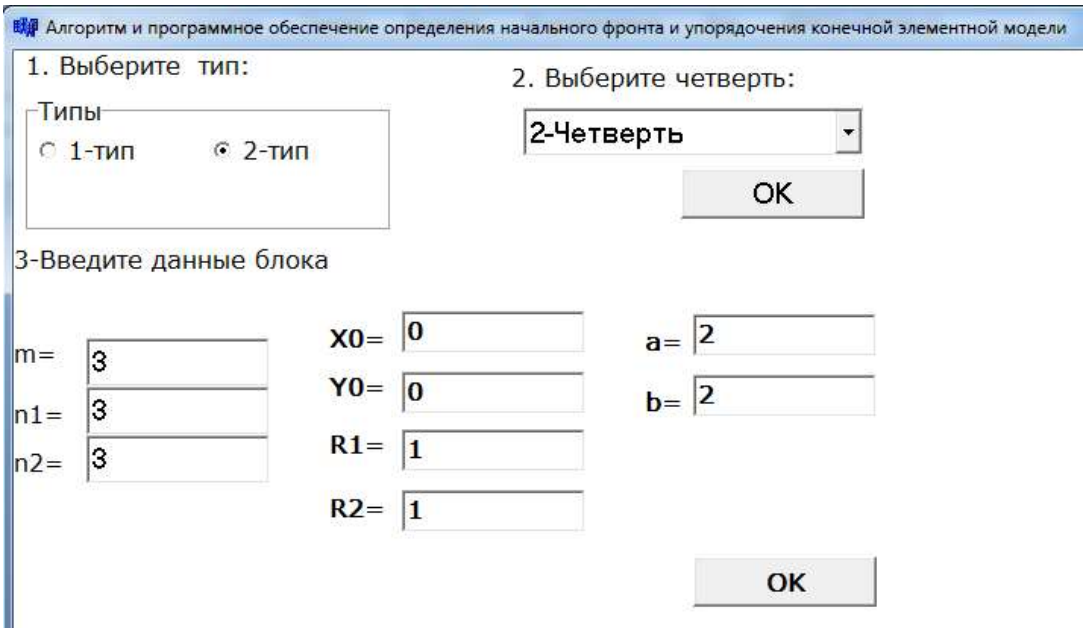


Рисунок 14. Часть интерфейса

Далее нам необходимо нажать клавишу "Сшивание", что бы наши подобласти объединились. В результате мы имеем:

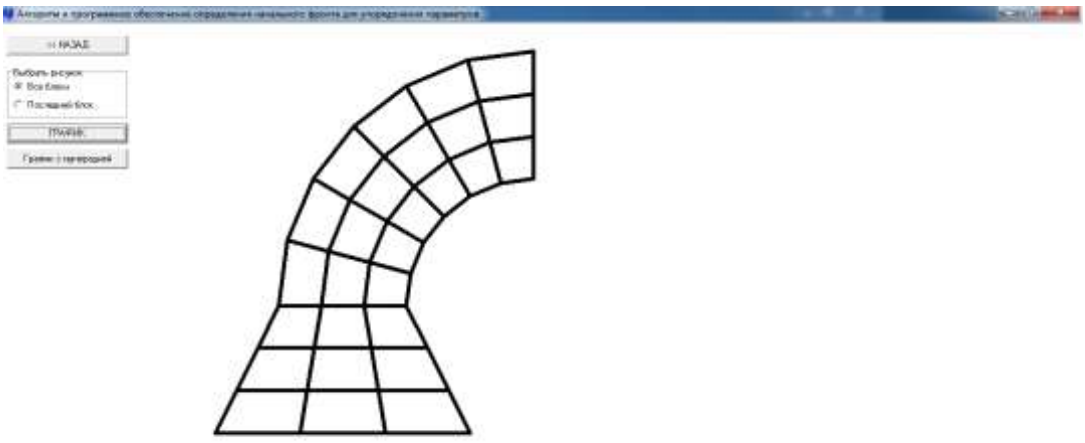


Рисунок 15. Дискретная модель

Затем вводим данные для третьей подобласти:

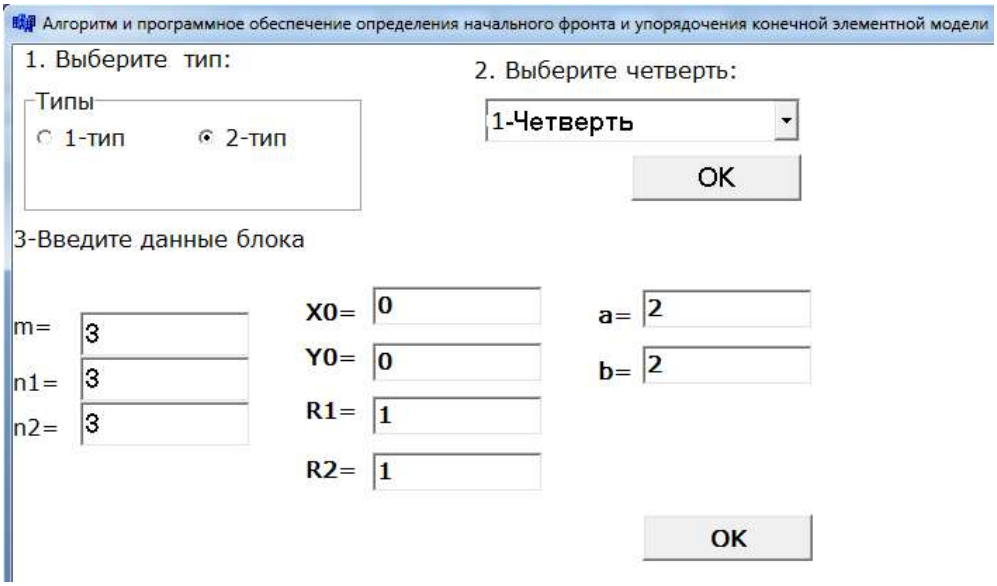


Рисунок 16. Часть интерфейса

И у нас уже три объединенные подобласти:

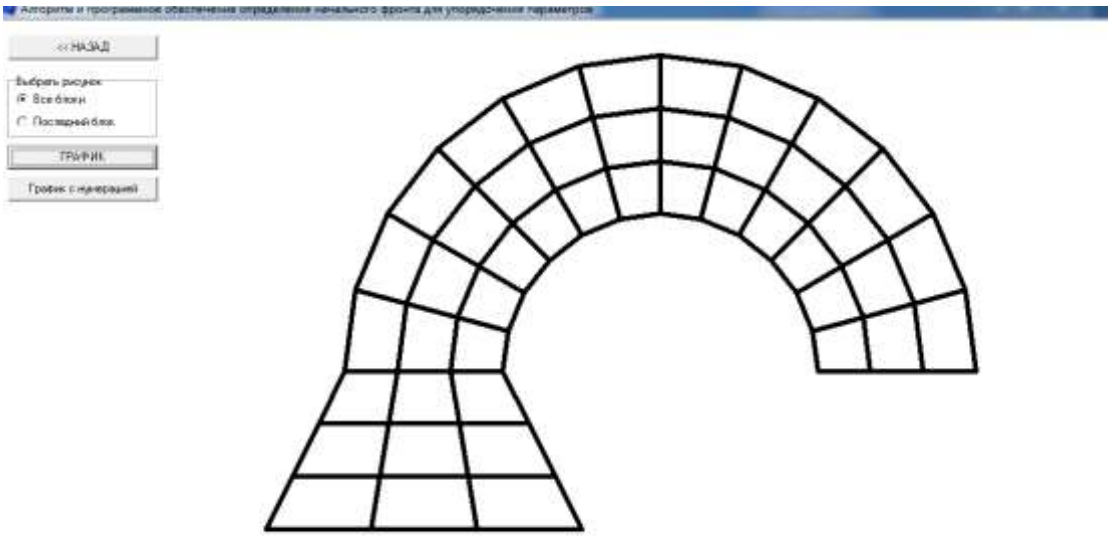


Рисунок 17. Дискретная модель

Аналогично вводим данные для четвертой нашей подобласти:

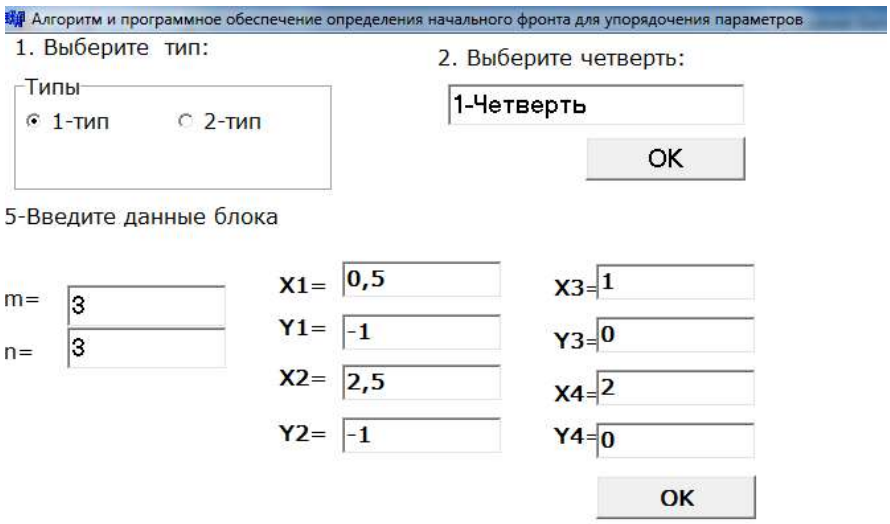


Рисунок 18. Часть интерфейса

и получаем:

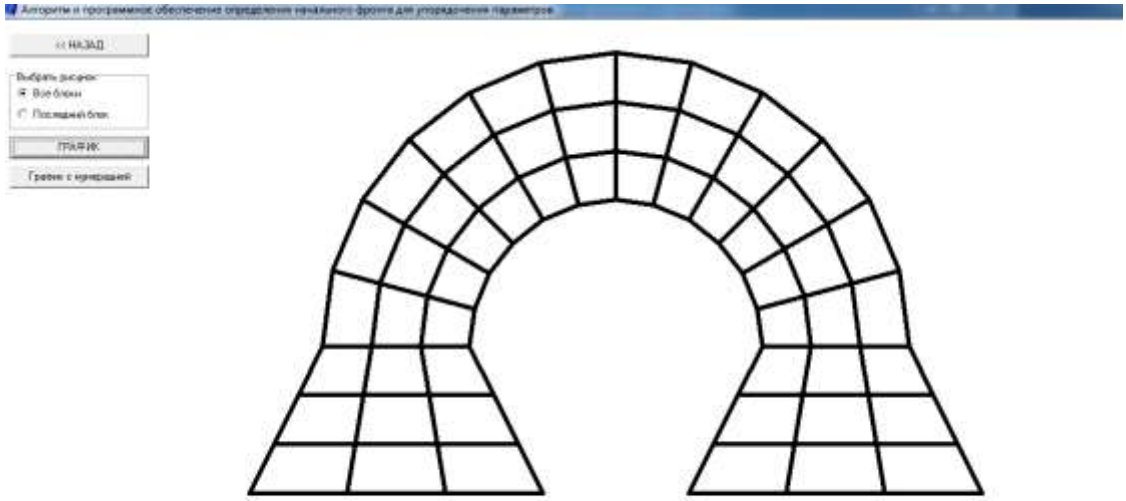


Рисунок 19. Дискретная модель

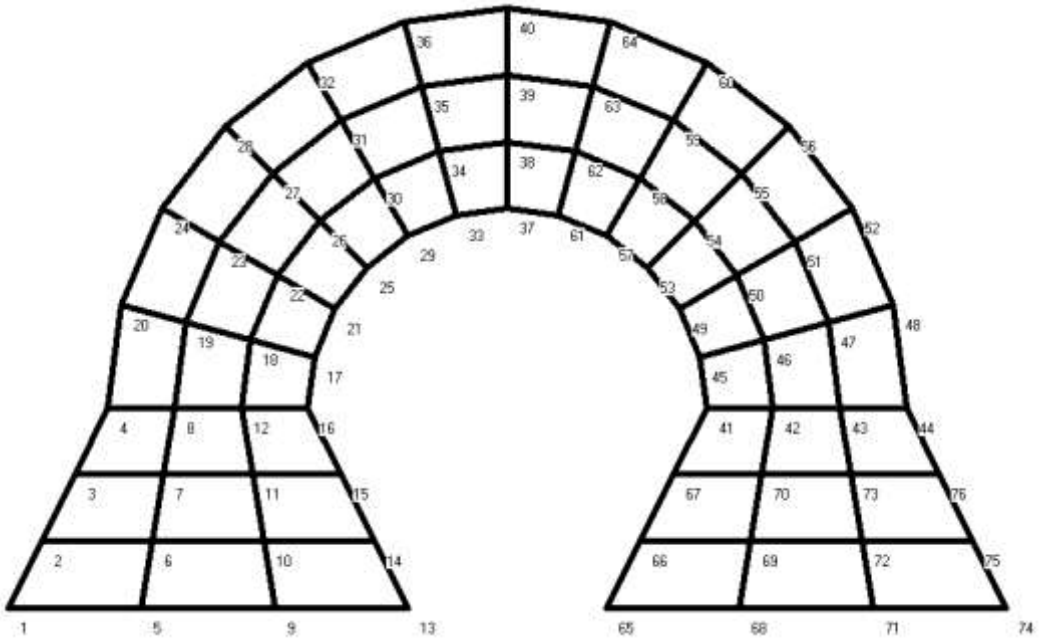
Общие результаты выглядят следующим образом:

Информация о блоках :

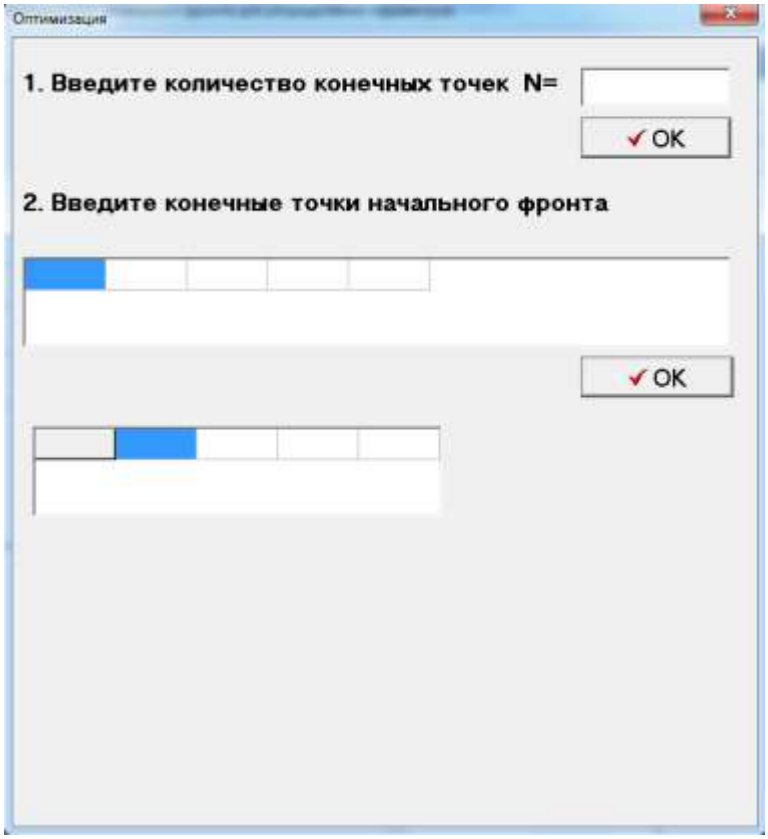
```
Блок № 4
Узловые точки последнего блока=16
Количество элементов последнего блока=9
Этот блок  добавляется к предыдущему
Совпадающие узловые точки блоков
41  42  43  44
13  14  15  16
Общее количество узлов блока >> 76
Количество элементов >> 54
------------------------------------------------------------
```

MN:

1	1	2	5	6
2	2	3	6	7
3	3	4	7	8
4	5	6	9	10
5	6	7	10	11
6	7	8	11	12
7	9	10	13	14
8	10	11	14	15
9	11	12	15	16
10	16	15	17	18
11	15	14	18	19
12	14	13	19	20
13	17	18	21	22
14	18	19	22	23
15	19	20	23	24
16	21	22	25	26
17	22	23	26	27
18	23	24	27	28
19	25	26	29	30
20	26	27	30	31
21	27	28	31	32



Здесь можем увидеть что l=16; Цель нашей работы минимизировать число l. Для этого нажимаем кнопку “ОПТИМИЗАЦИЯ”. На экран выводится следующий интерфейс:

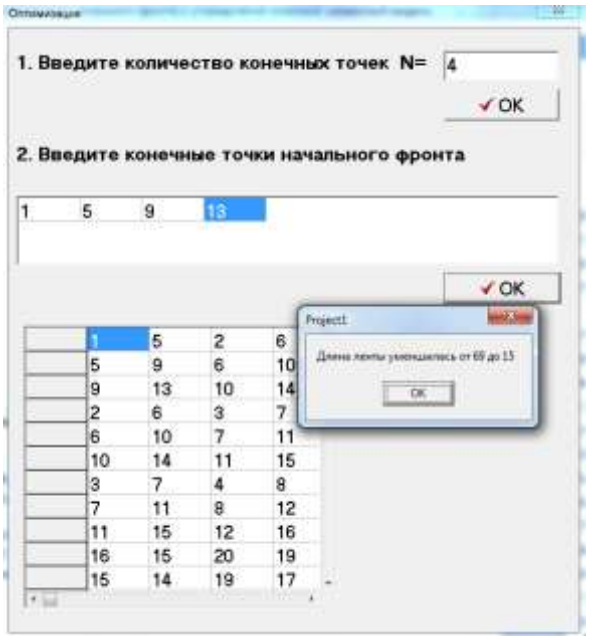


Вводим данные:

1. N=4;
2. 1 5 9 13;

Нажмем кнопку ОК.

После этого на экране появится результат после минимизации:



Ссравним результаты:

До:

1	2	5	6
2	3	6	7
3	4	7	8
5	6	9	10
6	7	10	11
7	8	11	12
9	10	13	14
10	11	14	15
11	12	15	16
16	12	17	18
12	8	18	19

После:

1	5	2	6
5	9	6	10
9	13	10	14
2	6	3	7
6	10	7	11
10	14	11	15
3	7	4	8
7	11	8	12
11	15	12	16
16	15	20	19
15	14	19	17

Число l уменьшилась от 69 до 15.

Заключение

В дипломной работе описана технология составления дискретной модели сложной двухмерной области. Исходная область разбивается на элементарные подобласти и для этих областей разработан алгоритм, позволяющий сшивание данных подобластей. На языке C++ написана программа с использованием методов и алгоритмов.

Для поставленной задачи изучены методы разбиения элементарных областей. В дипломной работе описаны 2 элементарные подобласти с подробными описаниями и иллюстрациями их алгоритмов. А также я изучил алгоритм сшивания подобластей и с помощью этого алгоритма составил программу. В итоге, имея все необходимые знания, я могу исследовать любую сложную фигуру, разбить их на подобластей, и дискретизировать.

Литература

1. Бьерн Страуструп Язык программирования C++: - Москва, Бином, 2011.
2. Работнов Ю.Н. Механика деформируемого твердого тела: - М.: Наука, 1988.
3. Норри Д. Фриз Ж. - Введение в метод конечных элементов: Мир
4. Розин Л.А. Стержневые системы как системы конечных элементов. Л.: Изд-во ЛГУ, 1976. Стр. 232
5. Стренг Г., Фикс Дж. Теория метода конечных элементов. М.: Мир, 1977. Стр. 349
6. Полатов А.М., Фёдоров А.Ю. Алгоритм минимизации ширины ленты системы уравнений // Современные информационные технологии в науке, образовании и практике. РФ, Оренбург, 2007. Стр. 103-105.

Приложение

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include<stdio.h>
#include<math.h>
#define  fayl   "umumiy.txt"
#define  fayl1   "oxirgi.txt"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
typedef float Mas[2][1000];
typedef int mas[4][1000];
typedef float mass[1000];
typedef float XY[2][4];
int tur,kurinish;
int Us=0,Es=0;
FILE *f;
float x0,y0,r1,r2,a,b;
int kb,n,m,ch,n1,n2,Us1,Es1,bloks_nomer=0;
Mas HK,HK1;
mas HN,HN1;
mass ai,bi;
XY xy;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
 Form1->WindowState=wsMaximized;
 Button1->Enabled=false;
 //Button2->Visible=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
 kurinish=ComboBox1->ItemIndex+1;
   Label18->Visible=true;
   Edit10->Visible=true;
   Label22->Visible=true;
   Label21->Visible=true;
   Edit11->Visible=true;
   Edit12->Visible=true;
```

```
switch(tur)
{
 case 1:
  Label1->Caption="n=";
  Label18->Visible=false;
  Edit10->Visible=false;
  Label16->Caption="X1=";
  Label15->Caption="Y1=";
  Label4->Caption="X2=";
  Label12->Caption="Y2=";
  Label14->Caption="X3=";
  Label13->Caption="Y3=";
  Label22->Caption="X4=";
  Label21->Caption="Y4=";
  break;
 case 2:
 case 3:
  Label1->Caption="n1=";
  Label18->Caption="n2=";
  Label16->Caption="X0=";
  Label15->Caption="Y0=";
  Label4->Caption="R1=";
  Label12->Caption="R2=";
  Label14->Caption="a=";
  Label13->Caption="b=";
  Label22->Visible=false;
  Label21->Visible=false;
  Edit11->Visible=false;
  Edit12->Visible=false;


  break;
 case 4:
  Label1->Caption="n1=";
  Label18->Caption="n2=";
  //Edit10->Visible=false;
  Label16->Caption="X0=";
  Label15->Caption="Y0=";
  Label4->Caption="R1=";
  Label12->Caption="R2=";
  Label14->Caption="a=";
  Label13->Caption="b=";
  Label22->Visible=false;
  Label21->Visible=false;
  Edit11->Visible=false;
  Edit12->Visible=false;


  break;
}
for (int i = 0; i < StringGrid1->RowCount; i++)
 {
```

```
    for (int j = 0; j < StringGrid1->ColCount; j++)
    {
      StringGrid1->Cells[j][i] = "1";
      StringGrid2->Cells[j][i] = "1";
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::RadioGroup1Click(TObject *Sender)
{
  tur=StrToInt(RadioGroup1->ItemIndex)+1;
  if(tur==1) ComboBox1->Style=csSimple;
  else ComboBox1->Style=csDropDownList;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
 bloks_nomer++;
  Memo1->Lines->Add("-----------------------------------------------------");
  Memo1->Lines->Add("Блок № "+IntToStr(bloks_nomer));
 switch(tur)
 {
   case 1:
    xy[0][0]=StrToFloat(Edit8->Text);
    xy[1][0]=StrToFloat(Edit9->Text);
    xy[0][1]=StrToFloat(Edit4->Text);
    xy[1][1]=StrToFloat(Edit5->Text);
    xy[0][2]=StrToFloat(Edit6->Text);
    xy[1][2]=StrToFloat(Edit7->Text);
    xy[0][3]=StrToFloat(Edit11->Text);
    xy[1][3]=StrToFloat(Edit12->Text);
    for(int i=0;i<m;i++) ai[i]=StrToFloat(StringGrid1->Cells[i][0]);
    for(int i=0;i<n;i++) bi[i]=StrToFloat(StringGrid2->Cells[i][0]);
    Us1=(n+1)*(m+1);
    Es1=n*m;
    if(bloks_nomer==1)
    {
    Us=(m+1)*(n+1);
    Es=m*n;
    rblok3(HK,HN,xy,ai,bi,m,n);
    }
    else{
    Button1->Enabled=true;
    rblok3(HK1,HN1,xy,ai,bi,m,n);
    }
    break;
   case 2:
   case 3:
    x0=StrToFloat(Edit8->Text);
    y0=StrToFloat(Edit9->Text);
    r1=StrToFloat(Edit4->Text);
    r2=StrToFloat(Edit5->Text);
```

```
  a=StrToFloat(Edit6->Text);
  b=StrToFloat(Edit7->Text);
  for(int i=0;i<m;i++) ai[i]=StrToFloat(StringGrid1->Cells[i][0]);
  Us1=(n1+n2+1)*(m+1);
  Es1=(n1+n2)*m;
  if(bloks_nomer==1)
  {
  Us=(n1+n2+1)*(m+1);
  Es=(n1+n2)*m;
  if(tur==2)
  rblok5(HK,HN,ai,n1,n2,m,r1,r2,a,b,x0,y0,kurinish);
  else
  rblok(HK,HN,ai,n1,n2,m,r1,r2,a,b,x0,y0,kurinish);
  }
  else
{
 Button1->Enabled=true;
// if(bloks_nomer>2)X0=X0+2*a;
 //Button2->Enabled=true;
 if(tur==2)
 rblok5(HK1,HN1,ai,n1,n2,m,r1,r2,a,b,x0,y0,kurinish);
 else
 rblok(HK1,HN1,ai,n1,n2,m,r1,r2,a,b,x0,y0,kurinish);
}
  break;
  case 4:
  x0=StrToFloat(Edit8->Text);
  y0=StrToFloat(Edit9->Text);
  r1=StrToFloat(Edit4->Text);
  r2=StrToFloat(Edit5->Text);
  a=StrToFloat(Edit6->Text);
  b=StrToFloat(Edit7->Text);
  for(int i=0;i<m;i++) ai[i]=StrToFloat(StringGrid1->Cells[i][0]);
  Us1=(n1+n2+1)*m+(n1+1)*(n2+1);
  Es1=(n1+n2)*m+n1*n2;
  if(bloks_nomer==1)
  {
   Us=(n1+n2+1)*m+(n1+1)*(n2+1);
  Es=(n1+n2)*m+n1*n2;
  rblok4(HK,HN,ai,n1,n2,m,r1,r2,a,b,x0,y0,kurinish);
  }
  else{
   Button1->Enabled=true;
   rblok4(HK1,HN1,ai,n1,n2,m,r1,r2,a,b,x0,y0,kurinish);
   }
  break;
}
Memo1->Lines->Add("Узловые точки последнего блока="+IntToStr(Us1));
Memo1->Lines->Add("Количество элементов последнего блока="+IntToStr(Es1));
if(bloks_nomer==1)Memo1->Lines->Add("                                ");
Label10->Caption=IntToStr(bloks_nomer+1)+"-Введите данные блока";//
```

```
int rowCount = StringGrid4->RowCount;
int colCount = StringGrid4->ColCount - 1;

int m1[1000];
int m2[1000];
int m4[1000];

int im1 = 0;
int im2 = 0;
int im4 = 0;

for (int row = 0; row < StringGrid4->RowCount; row++)
{
  for (int col = 1; col < StringGrid4->ColCount; col++)
  {
    int son = StrToInt(StringGrid4->Cells[col][row]);
    int findCount = 0;
    for (int irow = 0; irow < StringGrid4->RowCount; irow++)
    {
      for (int jcol = 1; jcol < StringGrid4->ColCount; jcol++)
      {
        if (son == StrToInt(StringGrid4->Cells[jcol][irow]))
        {
          findCount++;
        }
      }
    }

    if (findCount == 1)
    {
      bool find = false;
      for (int z = 0; z <im1; z++)
      {
        if (m1[z] == son)
        {
          find = true;
          break;
        }
      }

      if (!find)
      {
        m1[im1++] = son;
      }
    }
    else if (findCount == 2)
    {
      bool find = false;
      for (int z = 0; z <im2; z++)
      {
        if (m2[z] == son)
        {
```

```
            find = true;
            break;
          }
        }

        if (!find)
        {
          m2[im2++] = son;
        }
      }
      else if (findCount == 4)
      {
        bool find = false;
        for (int z = 0; z <im4; z++)
        {
          if (m4[z] == son)
          {
            find = true;
            break;
          }
        }

        if (!find)
        {
          m4[im4++] = son;
        }
      }
    }
  }

  if (im1 > 0)
  {
    Memo1->Lines->Add("Вершины: ");
    String s = "";
    for (int i = 0; i < im1; i++)
    {
      s += " " + IntToStr(m1[i]) + ",";
    }

    s +="  Количество="+IntToStr(im1);

    Memo1->Lines->Add(s);
    }

  if (im2 > 0)
  {
    Memo1->Lines->Add("Крайные точки: ");
    String s = "";
    for (int i = 0; i < im2; i++)
    {
       s += " " + IntToStr(m2[i]) +",";
    }
```

```
      s +="  Количество="+IntToStr(im2);
      Memo1->Lines->Add(s);
    }

    if (im4 > 0)
    {
      Memo1->Lines->Add("Внутренные точки: ");
      String s = "";
      for (int i = 0; i < im4; i++)
      {
        s += " " + IntToStr(m4[i]) + ",";
      }
      s +="  Количество=" +  IntToStr(im4);
      Memo1->Lines->Add(s);
      Memo1->Lines->Add("----------------------------------------------------");

    }


    int z = 10;
    z = 15;
 }


//---------------------------------------------------------------------------
void TForm1::rblok5(Mas HKy,mas HNy,mass ai,int n1,int n2,int m,float r1,float r2,float a,float
b,
float x,float y,int tur)
{
/* int  S=0,S1;
 float alfa,r;
for(int i=0;i<m;i++) S+=ai[i];
for(int i=0;i<=(n1+n2);i++)
{
 S1=0;
 if(i<=n1) alfa=float(i)/n1*atan(b/a);
 else alfa=atan(b/a)+float(i-n1)/n2*atan(a/b);
 if(i!=(n1+n2)) r=r1*r2/(cos(alfa)*pow(r2*r2+r1*r1*tan(alfa)*tan(alfa),0.5));
 else r=r2;
 for(int j=0;j<=m;j++)
 {
  if(j>0)S1+=ai[j-1];
  if(j==0)
   {
   HKy[0][i*(m+1)+j]=r*cos(alfa)+x;
   HKy[1][i*(m+1)+j]=r*sin(alfa)+y;
   }
   else
   {
  if(i<=n1)
   {
    HKy[0][i*(m+1)+j]=((a/cos(alfa)-r)*S1/S+r)*cos(alfa)+x;
```

```
    HKy[1][i*(m+1)+j]=((a/cos(alfa)-r)*S1/S+r)*sin(alfa)+y;
   }
  else
   {
    HKy[0][i*(m+1)+j]=((b/sin(alfa)-r)*S1/S+r)*cos(alfa)+x;
    HKy[1][i*(m+1)+j]=((b/sin(alfa)-r)*S1/S+r)*sin(alfa)+y;
   }
 }
 }
}
for(int i=0;i<(n1+n2+1)*(m+1);i++)
{
   if(tur==2)
     HKy[0][i]=2*x-HKy[0][i];
   if(tur==3)
    {
     HKy[0][i]=2*x-HKy[0][i];
     HKy[1][i]=2*y-HKy[1][i];
    }
   if(tur==4)
     HKy[1][i]=2*y-HKy[1][i];
}*/
 int  S=0,S1;
 float alfa,r,A,B;
for(int i=0;i<m;i++) S+=ai[i];
for(int i=0;i<=(n1+n2);i++)
{
 S1=0;
 if(i<=n1) alfa=float(i)/n1*atan(b/a);
 else alfa=atan(b/a)+float(i-n1)/n2*atan(a/b);
 if(i!=(n1+n2)) r=r1*r2/(cos(alfa)*pow(r2*r2+r1*r1*tan(alfa)*tan(alfa),0.5));
 else r=r2;
 for(int j=0;j<=m;j++)
 {
  if(j>0)S1+=ai[j-1];
  if(j==0)
   {

   HKy[0][i*(m+1)+j]=r*cos(alfa)+x;
   HKy[1][i*(m+1)+j]=r*sin(alfa)+y;
   if(i<=n1)
   { A=x+a;
   B=(b)/n1*i+y;
   }
   else
   {
   A=(a)/n2*(n2+n1-i)+x;
   B=y+b;
   }

   }
   else
```

```
  {
 if(i<=n1)
  {
  HKy[0][i*(m+1)+j]=r*cos(alfa)+x+(A-r*cos(alfa)-x)*S1/S;
  HKy[1][i*(m+1)+j]=r*sin(alfa)+y+(B-r*sin(alfa)-y)*S1/S;
  // HKy[0][i*(m+1)+j]=((a/cos(alfa)-r)*S1/S+r)*cos(alfa)+x;
   //HKy[1][i*(m+1)+j]=((a/cos(alfa)-r)*S1/S+r)*sin(alfa)+y;
  }
 else
  {
   HKy[0][i*(m+1)+j]=r*cos(alfa)+x+(A-r*cos(alfa)-x)*S1/S;
    HKy[1][i*(m+1)+j]=r*sin(alfa)+y+(B-r*sin(alfa)-y)*S1/S;

  // HKy[0][i*(m+1)+j]=((b/sin(alfa)-r)*S1/S+r)*cos(alfa)+x;
   //HKy[1][i*(m+1)+j]=((b/sin(alfa)-r)*S1/S+r)*sin(alfa)+y;
  }
 }
 }
}
for(int i=0;i<(n1+n2+1)*(m+1);i++)
{
  if(tur==2)
    HKy[0][i]=2*x-HKy[0][i];
  if(tur==3)
   {
    HKy[0][i]=2*x-HKy[0][i];
    HKy[1][i]=2*y-HKy[1][i];
   }
  if(tur==4)
    HKy[1][i]=2*y-HKy[1][i];
}
int k=1;
for(int i=1;i<=(n1+n2)*m;i++)
{
 HNy[0][i-1]=k;
 HNy[1][i-1]=k+1;
 HNy[2][i-1]=HNy[0][i-1]+m+1;
 HNy[3][i-1]=HNy[0][i-1]+m+2;
 if(i%m==0)k+=2;
 else k++;
}
StringGrid3->RowCount=(n1+n2+1)*(m+1);
StringGrid4->RowCount=(n1+n2)*m;
for(int i=1;i<=(n1+n2+1)*(m+1);i++)
 {
  StringGrid3->Cells[0][i-1]=i;
  StringGrid3->Cells[1][i-1]=FloatToStrF(HKy[0][i-1],ffFixed,8,2);
  StringGrid3->Cells[2][i-1]=FloatToStrF(HKy[1][i-1],ffFixed,8,2);
 }
 for(int i=1;i<=(n1+n2)*m;i++)
 {
  StringGrid4->Cells[0][i-1]=i;
```

```
  StringGrid4->Cells[1][i-1]=HNy[0][i-1];
  StringGrid4->Cells[2][i-1]=HNy[1][i-1];
  StringGrid4->Cells[3][i-1]=HNy[2][i-1];
  StringGrid4->Cells[4][i-1]=HNy[3][i-1];
 }
}
void TForm1::rblok(Mas HKy,mas HNy,mass ai,int n1,int n2,int m,float r1,float r2,float a,float
b,
float x,float y,int tur)
{
  float R1,R2;
  float S=0;
 float alfa,r;
for(int i=0;i<m;i++) S+=ai[i];
for(int i=0;i<=n1+n2;i++)
{
 if(i<=n1) alfa=float(i)/n1*atan(b/a);
 else alfa=atan(b/a)+float(i-n1)/n2*atan(a/b);
 R1=r1;
 R2=r2;
 for(int j=0;j<=m;j++)
 {
  if(fabs(alfa-M_PI/2)<=0.001) r=R2;
  else r=R1*R2/(cos(alfa)*pow(R2*R2+R1*R1*tan(alfa)*tan(alfa),0.5));
  HKy[0][i*(m+1)+j]=r*cos(alfa)+x;
  HKy[1][i*(m+1)+j]=r*sin(alfa)+y;
  R1+=(a-r1)*ai[j]/S;
  R2+=(b-r2)*ai[j]/S;
 }
}
for(int i=0;i<(n1+n2+1)*(m+1);i++)
{
   if(tur==2)
     HKy[0][i]=2*x-HKy[0][i];
   if(tur==3)
    {
     HKy[0][i]=2*x-HKy[0][i];
     HKy[1][i]=2*y-HKy[1][i];
    }
   if(tur==4)
     HKy[1][i]=2*y-HKy[1][i];
}
int k=1;
for(int i=1;i<=(n1+n2)*m;i++)
{
 HNy[0][i-1]=k;
 HNy[1][i-1]=k+1;
 HNy[2][i-1]=HNy[0][i-1]+m+1;
 HNy[3][i-1]=HNy[0][i-1]+m+2;
 if(i%m==0)k+=2;
 else k++;
}
```

```
StringGrid3->RowCount=(n1+n2+1)*(m+1);
StringGrid4->RowCount=(n1+n2)*m;
for(int i=1;i<=(n1+n2+1)*(m+1);i++)
 {
  StringGrid3->Cells[0][i-1]=i;
  StringGrid3->Cells[1][i-1]=FloatToStrF(HKy[0][i-1],ffFixed,8,2);
  StringGrid3->Cells[2][i-1]=FloatToStrF(HKy[1][i-1],ffFixed,8,2);
 }
 for(int i=1;i<=(n1+n2)*m;i++)
 {
  StringGrid4->Cells[0][i-1]=i;
  StringGrid4->Cells[1][i-1]=HNy[0][i-1];
  StringGrid4->Cells[2][i-1]=HNy[1][i-1];
  StringGrid4->Cells[3][i-1]=HNy[2][i-1];
  StringGrid4->Cells[4][i-1]=HNy[3][i-1];
 }


}
void TForm1::rblok3(Mas HKy,mas HNy,XY xy,mass ai,mass bi,int m,int n)
{
 float ax[2][100],ay[2][100],S1=0,S2=0,s=0;
for(int i=0;i<m;i++) S1+=ai[i];
for(int j=0;j<n;j++) S2+=bi[j];
 for(int i=0;i<=n;i++)
 {
  ax[0][i]=xy[0][0]+s*(xy[0][2]-xy[0][0])/S2;
  ax[1][i]=xy[0][1]+s*(xy[0][3]-xy[0][1])/S2;
  ay[0][i]=xy[1][0]+s*(xy[1][2]-xy[1][0])/S2;
  ay[1][i]=xy[1][1]+s*(xy[1][3]-xy[1][1])/S2;
  s+=bi[i];
 }
 s=0;
 for(int i=0;i<=n;i++)
  {
    s=0;
   for(int j=0;j<=m;j++)
    {
     HKy[0][i*(m+1)+j]=ax[0][i]+s*(ax[1][i]-ax[0][i])/S1;
     HKy[1][i*(m+1)+j]=ay[0][i]+s*(ay[1][i]-ay[0][i])/S1;
     s+=ai[j];
    }
  }
  int k=1;
for(int i=1;i<=n*m;i++)
{
 HNy[0][i-1]=k;
 HNy[1][i-1]=k+1;
 HNy[2][i-1]=HNy[0][i-1]+m+1;
 HNy[3][i-1]=HNy[0][i-1]+m+2;
 if(i%m==0)k+=2;
 else k++;
```

```
}
StringGrid3->RowCount=(n+1)*(m+1);
StringGrid4->RowCount=n*m;
for(int i=1;i<=(n+1)*(m+1);i++)
 {
  StringGrid3->Cells[0][i-1]=i;
  StringGrid3->Cells[1][i-1]=FloatToStrF(HKy[0][i-1],ffFixed,8,2);
  StringGrid3->Cells[2][i-1]=FloatToStrF(HKy[1][i-1],ffFixed,8,2);
 }
 for(int i=1;i<=n*m;i++)
 {
  StringGrid4->Cells[0][i-1]=i;
  StringGrid4->Cells[1][i-1]=HNy[0][i-1];
  StringGrid4->Cells[2][i-1]=HNy[1][i-1];
  StringGrid4->Cells[3][i-1]=HNy[2][i-1];
  StringGrid4->Cells[4][i-1]=HNy[3][i-1];
 }
}
void TForm1::pernom_p(Mas HKy,mas HNy,Mas HKy1,mas HNy1,int &Us,int &Es,int Us1,int
Es1)
{
 int mos[100],mos1[100],k=0,t=0,H[1000],i,j;
 bool p=false;
for(i=1;i<=Us;i++){
  for(j=1;j<=Us1;j++){
  if(fabs(HKy[0][i-1]-HKy1[0][j-1])<0.001&&fabs(HKy[1][i-1]-HKy1[1][j-1])<0.001
  )
    {
      p=true;     break;
   }
}
if(p){
mos[k]=i;
mos1[k]=j;
k++;
}
p=false;
}
t=0;
p=true;
for(i=1;i<=Us1;i++){
 for(j=0;j<k;j++){
  if(i==mos1[j]){
    p=false;
    H[i-1]=mos[j];
    }
  if(i>mos1[j])t++;
}
if(p){
 H[i-1]=Us+i-t;
 HKy[0][Us+i-1-t]=HKy1[0][i-1];
 HKy[1][Us+i-1-t]=HKy1[1][i-1];
```

```
}
t=0;
p=true;
}
for(i=1;i<=Es1;i++)
{
 HNy[0][Es+i-1]=H[HNy1[0][i-1]-1];
 HNy[1][Es+i-1]=H[HNy1[1][i-1]-1];
 HNy[2][Es+i-1]=H[HNy1[2][i-1]-1];
 HNy[3][Es+i-1]=H[HNy1[3][i-1]-1];
}
Us+=Us1-k;
Es+=Es1;
Memo1->Lines->Add("Этот блок  добавляется к предыдущему");   //Bu blok avvalgilariga
qo'shiladi
Memo1->Lines->Add("Совпадающие узловые точки блоков");  //Bloklarning mos kelgan
nuqtalari
AnsiString S1="",S2="";
for(int i=0;i<k;i++)
 S1+=IntToStr(mos[i])+"  ";
for(int i=0;i<k;i++)
 S2+=IntToStr(mos1[i])+"  ";
 //Memo1->Lines->Add(S1);
 Memo1->Lines->Add(S2);
Memo1->Lines->Add("Общее количество узлов блока >> "+IntToStr(Us));  //Bloklarning
umumiy tugun nuqtalari soni
Memo1->Lines->Add("Количество элементов >> "+IntToStr(Es));//Elementlari soni
StringGrid3->RowCount=Us;
StringGrid4->RowCount=Es;
for(int i=1;i<=Us;i++)
 {
  StringGrid3->Cells[0][i-1]=i;
  StringGrid3->Cells[1][i-1]=FloatToStrF(HKy[0][i-1],ffFixed,8,2);
  StringGrid3->Cells[2][i-1]=FloatToStrF(HKy[1][i-1],ffFixed,8,2);
 }
 for(int i=1;i<=Es;i++)
 {
  StringGrid4->Cells[0][i-1]=i;
  StringGrid4->Cells[1][i-1]=HNy[0][i-1];
  StringGrid4->Cells[2][i-1]=HNy[1][i-1];
  StringGrid4->Cells[3][i-1]=HNy[2][i-1];
  StringGrid4->Cells[4][i-1]=HNy[3][i-1];
 }
}
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
  switch(tur)
  {
   case 1:
    m=StrToInt(Edit1->Text);
    n=StrToInt(Edit2->Text);
    StringGrid1->ColCount=m;
```

```
    StringGrid2->ColCount=n;
    StringGrid1->Col=0;
    StringGrid1->Row=0;
    StringGrid2->Col=0;
    StringGrid2->Row=0;
    break;
   case 2:
   case 3:
    m=StrToInt(Edit1->Text);
    n1=StrToInt(Edit2->Text);
    n2=StrToInt(Edit10->Text);
    StringGrid1->ColCount=m;
    StringGrid1->Col=0;
    StringGrid1->Row=0;
    break;
   case 4:
    m=StrToInt(Edit1->Text);
    n1=StrToInt(Edit2->Text);
    n2=StrToInt(Edit10->Text);
    StringGrid1->ColCount=m;
    StringGrid1->Col=0;
    StringGrid1->Row=0;
    break;
  }
  SpeedButton2Click(SpeedButton1);
  }
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  pernom_p(HK,HN,HK1,HN1,Us,Es,Us1,Es1);
  int rowCount = StringGrid4->RowCount;
   int colCount = StringGrid4->ColCount - 1;

   int m1[1000];
   int m2[1000];
   int m4[1000];

   int im1 = 0;
   int im2 = 0;
   int im4 = 0;

   for (int row = 0; row < StringGrid4->RowCount; row++)
   {
      for (int col = 1; col < StringGrid4->ColCount; col++)
      {
         int son = StrToInt(StringGrid4->Cells[col][row]);
         int findCount = 0;
         for (int irow = 0; irow < StringGrid4->RowCount; irow++)
         {
            for (int jcol = 1; jcol < StringGrid4->ColCount; jcol++)
            {
```

```
      if (son == StrToInt(StringGrid4->Cells[jcol][irow]))
      {
        findCount++;
      }
    }
  }

  if (findCount == 1)
  {
    bool find = false;
    for (int z = 0; z <im1; z++)
    {
      if (m1[z] == son)
      {
        find = true;
        break;
      }
    }

    if (!find)
    {
      m1[im1++] = son;
    }
  }
  else if (findCount == 2)
  {
    bool find = false;
    for (int z = 0; z <im2; z++)
    {
      if (m2[z] == son)
      {
        find = true;
        break;
      }
    }

    if (!find)
    {
      m2[im2++] = son;
    }
  }
  else if (findCount == 4)
  {
    bool find = false;
    for (int z = 0; z <im4; z++)
    {
      if (m4[z] == son)
      {
        find = true;
        break;
      }
    }
```

```
        if (!find)
        {
          m4[im4++] = son;
        }
      }
    }
  }

  if (im1 > 0)
  {
    Memo1->Lines->Add("Вершины: ");
    String s = "";
    for (int i = 0; i < im1; i++)
    {
      s += " " + IntToStr(m1[i]) + ",";
    }

    s +="  Количество="+IntToStr(im1);

    Memo1->Lines->Add(s);
  }

  if (im2 > 0)
  {
    Memo1->Lines->Add("Крайные точки: ");
    String s = "";
    for (int i = 0; i < im2; i++)
    {
       s += " " + IntToStr(m2[i]) +",";
    }
    s +="  Количество="+IntToStr(im2);
    Memo1->Lines->Add(s);
  }

  if (im4 > 0)
  {
    Memo1->Lines->Add("Внутренные точки: ");
    String s = "";
    for (int i = 0; i < im4; i++)
    {
      s += " " + IntToStr(m4[i]) + ",";
    }
    s +="  Количество=" +  IntToStr(im4);
    Memo1->Lines->Add(s);
  }


  int z = 10;
  z = 15;

Memo1->Lines->Add("-----------------------------------------------------");
```

```
}
//---------------------------------------------------------------------------

void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
f=fopen(fayl,"w+");
  fprintf(f,"%3d       %3d",Us,Es);
  fprintf(f,"\n");
  for(int i=1;i<=Es;i++)
  {
   for(int j=1;j<=4;j++)
   {
    fprintf(f,"%3d     ",HN[j-1][i-1]);
   }
   fprintf(f,"\n");
  }
  for(int i=1;i<=Us;i++)
  {
   for(int j=1;j<=2;j++)
    fprintf(f,"%3f    ",HK[j-1][i-1]);
    fprintf(f,"\n");
  }
  fclose(f);
f=fopen(fayl1,"w+");
  fprintf(f,"%3d       %3d",Us1,Es1);
  fprintf(f,"\n");
  for(int i=1;i<=Es1;i++)
  {
   for(int j=1;j<=4;j++)
   {
    fprintf(f,"%3d     ",HN1[j-1][i-1]);
   }
   fprintf(f,"\n");
  }
  for(int i=1;i<=Us1;i++)
  {
   for(int j=1;j<=2;j++)
    fprintf(f,"%3f    ",HK1[j-1][i-1]);
    fprintf(f,"\n");
  }
  fclose(f);
  Form2->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
   f=fopen(fayl,"w+");
   fprintf(f,"%3d       %3d",Us,Es);
   fprintf(f,"\n");

   for(int i=1;i<=Es;i++)
```

```
  {
    for(int j=1;j<=4;j++)
    {
      fprintf(f,"%3d     ",HN[j-1][i-1]);
    }
    fprintf(f,"\n");
  }

  for(int i=1;i<=Us;i++)
  {
    for(int j=1;j<=2;j++)
      fprintf(f,"%3f    ",HK[j-1][i-1]);
    fprintf(f,"\n");
  }

  fclose(f);
  Form3->ShowModal();
}
//---------------------------------------------------------------------------
void TForm1::rblok4(Mas HKy,mas HNy,mass ai,int n1,int n2,int m,float r1,float r2,float a,float
b,
float x,float y,int tur)
{
  float R1,R2;
  float S=0;
 float alfa,r,**rx=new float*[2],**lx=new float*[2];
 int Us=(n1+n2+1)*(m+1),Es=(n1+n2)*m;
for(int i=0;i<2;i++)
{
 lx[i]=new float[n1+n2+1];
 rx[i]=new float[n1+n2+1];
}
for(int i=0;i<m;i++) S+=ai[i];
for(int i=0;i<=n1+n2;i++)
{
 if(i<=n1) alfa=float(i)/n1*atan(b/a);
 else alfa=atan(b/a)+float(i-n1)/n2*atan(a/b);
 R1=a;
 R2=b;
 //for(int j=0;j<=m;j++)
 //{
  if(fabs(alfa-M_PI/2)<=0.001) r=R2;
  else r=R1*R2/(cos(alfa)*pow(R2*R2+R1*R1*tan(alfa)*tan(alfa),0.5));
 // HKy[0][i*(m+1)+j]=r*cos(alfa)+x;
 // HKy[1][i*(m+1)+j]=r*sin(alfa)+y;
 // R1+=(a-r1)*ai[j]/S;
 // R2+=(b-r2)*ai[j]/S;
 //}
 rx[0][i]=r*cos(alfa)+x;
 rx[1][i]=r*sin(alfa)+y;
}
int c=0;
```

```
for(c=0;c<=n2;c++)
{
 lx[0][c]=r1+x;
 lx[1][c]=r2*c/(n2+0.0)+y;
}
c--;
for(int i=n1;i>=0;i--)
{
 lx[0][c]=r1-(n1-i)/(n1+0.0)*r1+x;
 lx[1][c]=r2+y;
 c++;
}
for(int i=0;i<=n1+n2;i++)
{
  float S1=0;
 for(int j=0;j<=m;j++)
 {
  HKy[0][i*(m+1)+j]=lx[0][i]+(rx[0][i]-lx[0][i])*S1/S;
  HKy[1][i*(m+1)+j]=lx[1][i]+(rx[1][i]-lx[1][i])*S1/S;
  if(j<m) S1+=ai[j];
 }
}
int k=1;
for(int i=1;i<=Es;i++)
{
 HNy[0][i-1]=k;
 HNy[1][i-1]=k+1;
 HNy[2][i-1]=HNy[0][i-1]+m+1;
 HNy[3][i-1]=HNy[0][i-1]+m+2;
 if(i%m==0)k+=2;
 else k++;
}
/*
StringGrid3->RowCount=Us;
StringGrid4->RowCount=Es;
for(int i=1;i<=Us;i++)
 {
  StringGrid3->Cells[0][i-1]=i;
  StringGrid3->Cells[1][i-1]=FloatToStrF(HKy[0][i-1],ffFixed,8,2);
  StringGrid3->Cells[2][i-1]=FloatToStrF(HKy[1][i-1],ffFixed,8,2);
 }
 for(int i=1;i<=Es;i++)
 {
  StringGrid4->Cells[0][i-1]=i;
  StringGrid4->Cells[1][i-1]=HNy[0][i-1];
  StringGrid4->Cells[2][i-1]=HNy[1][i-1];
  StringGrid4->Cells[3][i-1]=HNy[2][i-1];
  StringGrid4->Cells[4][i-1]=HNy[3][i-1];
 }

 */
Mas HKy1;
```

```
mas HNy1;
int Us1=(n1+1)*(n2+1),Es1=n1*n2;
XY xy1;
xy1[0][0]=x;
xy1[1][0]=y;
xy1[0][1]=x+r1;
xy1[1][1]=y;
xy1[0][2]=x;
xy1[1][2]=y+r2;
xy1[0][3]=x+r1;
xy1[1][3]=y+r2;
mass ay,by;
for(int i=1;i<=n1;i++) ay[i-1]=1;
for(int i=1;i<=n2;i++) by[i-1]=1;
rblok3(HKy1,HNy1,xy1,ay,by,n1,n2);
pernom_p(HKy,HNy,HKy1,HNy1,Us,Es,Us1,Es1);
for(int i=0;i<Us;i++)
{
  if(tur==2)
    HKy[0][i]=2*x-HKy[0][i];
  if(tur==3)
   {
    HKy[0][i]=2*x-HKy[0][i];
    HKy[1][i]=2*y-HKy[1][i];
   }
  if(tur==4)
    HKy[1][i]=2*y-HKy[1][i];
}
StringGrid3->RowCount=Us;
StringGrid4->RowCount=Es;
for(int i=1;i<=Us;i++)
 {
  StringGrid3->Cells[0][i-1]=i;
  StringGrid3->Cells[1][i-1]=FloatToStrF(HKy[0][i-1],ffFixed,8,2);
  StringGrid3->Cells[2][i-1]=FloatToStrF(HKy[1][i-1],ffFixed,8,2);
 }
 for(int i=1;i<=Es;i++)
 {
  StringGrid4->Cells[0][i-1]=i;
  StringGrid4->Cells[1][i-1]=HNy[0][i-1];
  StringGrid4->Cells[2][i-1]=HNy[1][i-1];
  StringGrid4->Cells[3][i-1]=HNy[2][i-1];
  StringGrid4->Cells[4][i-1]=HNy[3][i-1];
 }
 }


void __fastcall TForm1::FormCreate(TObject *Sender)
{

}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include<stdio.h>
#include<math.h>
#define F  "umumiy.txt"
#define F1  "oxirgi.txt"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
float A,B,M,*HK[2],*HK1[2],maxx,maxy;
int Us,Es,Us1,Es1,*HN[4],*HN1[4],X_Pa,Y_Pa,X_Yu,Y_Yu;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button1Click(TObject *Sender)
{
 Button2->Enabled=true;
if(Es1==0||Us1==0) RadioGroup1->ItemIndex=0;
switch(RadioGroup1->ItemIndex)
{
 case 0: Chizish(HN,HK,Us,Es);break;
 case 1: Chizish(HN1,HK1,Us1,Es1);break;
}
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormShow(TObject *Sender)
{
 Button2->Enabled=false;
  Form2->WindowState=wsMaximized;
FILE *out;
out=fopen(F,"r+");
fscanf(out,"%d",&Us);
fscanf(out,"%d",&Es);
for(int i=1;i<=2;i++)
HK[i-1]=new float[Us];
for(int i=1;i<=4;i++)
HN[i-1]=new int[Es];
for(int i=1;i<=Es;i++)
 for(int j=1;j<=4;j++)
  fscanf(out,"%d",&HN[j-1][i-1]);
for(int i=1;i<=Us;i++)
 for(int j=1;j<=2;j++)
  fscanf(out,"%f",&HK[j-1][i-1]);
```

```
fclose(out);
out=fopen(F1,"r+");
fscanf(out,"%d",&Us1);
fscanf(out,"%d",&Es1);
for(int i=1;i<=2;i++)
HK1[i-1]=new float[Us1];
for(int i=1;i<=4;i++)
HN1[i-1]=new int[Es1];
for(int i=1;i<=Es1;i++)
 for(int j=1;j<=4;j++)
  fscanf(out,"%d",&HN1[j-1][i-1]);
for(int i=1;i<=Us1;i++)
 for(int j=1;j<=2;j++)
  fscanf(out,"%f",&HK1[j-1][i-1]);
fclose(out);
maxx=fabs(HK[0][0]);maxy=fabs(HK[1][0]);
for(int i=1;i<=Us;i++)
{
if(fabs(HK[0][i-1])>maxx)maxx=fabs(HK[0][i-1]);
if(fabs(HK[1][i-1])>maxy)maxy=fabs(HK[1][i-1]);
}
A=2*maxx;
B=2*maxy;
if(A/(Form2->Width-100)>B/(Form2->Height-100))M=(Form2->Width-100)/A;
else M=(Form2->Height-100)/B;
if(M<1){Edit1->Text="1 : "+FloatToStrF(1/M,ffFixed,8,2)+" KICHRAYTIRISH";}
else {Edit1->Text=FloatToStrF(M,ffFixed,8,2)+" : 1"+"KATTALASHTIRISH" ;}
}
//---------------------------------------------------------------------------
void __fastcall TForm2::BitBtn1Click(TObject *Sender)
{
 Close();
}
//---------------------------------------------------------------------------
void TForm2::Tugun_nuqta_koordinata(TCanvas *canvas,float **HKy,int Us)
{
int m;
/*
  int rowCount = Form1->StringGrid4->RowCount;
  int colCount = Form1->StringGrid4->ColCount - 1;

  int *massive = new int[2*rowCount * colCount];
  for (int col = 0; col < colCount; col++)
  {
     for (int row = 0; row < rowCount; row++)
     {
        massive[col + row]
     }
  }
*/

 for(int i=0;i<Us;i++)
```

```
  {
    canvas->TextOutA(X_Pa+int(M*HKy[0][i])+10,Y_Pa-int(M*HKy[1][i])+10,IntToStr(i+1));
  }
}

//---------------------------------------------------------------------------
void  TForm2::Chizish(int **HNy,float **HKy,int Us,int Es)
{
X_Pa=ClientWidth/2-10,Y_Pa=Form2->ClientHeight/2+5,X_Yu=Form2->ClientWidth-20,
Y_Yu=10;
//int X_Pa=20,Y_Pa=Form2->ClientHeight-20,X_Yu=Form2->ClientWidth-20,Y_Yu=10;
Image1->Canvas->Brush->Color=clWhite;
Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));
Image1->Canvas->Pen->Width=5;
/*Image1->Canvas->MoveTo(X_Pa,Y_Pa);
Image1->Canvas->LineTo(X_Pa,Y_Yu);
Image1->Canvas->MoveTo(X_Pa-3,Y_Yu+3);
Image1->Canvas->LineTo(X_Pa,Y_Yu);
Image1->Canvas->MoveTo(X_Pa+3,Y_Yu+3);
Image1->Canvas->LineTo(X_Pa,Y_Yu);
Image1->Canvas->MoveTo(X_Pa,Y_Pa);
Image1->Canvas->LineTo(X_Yu,Y_Pa);
Image1->Canvas->MoveTo(X_Yu-3,Y_Pa+3);
Image1->Canvas->LineTo(X_Yu,Y_Pa);
Image1->Canvas->MoveTo(X_Yu-3,Y_Pa-3);
Image1->Canvas->LineTo(X_Yu,Y_Pa);*/
for(int i=1;i<=Es;i++)
{
 Image1->Canvas->MoveTo(X_Pa+int(M*HKy[0][HNy[0][i-1]-1]),Y_Pa-
int(M*HKy[1][HNy[0][i-1]-1]));
 Image1->Canvas->LineTo(X_Pa+int(M*HKy[0][HNy[1][i-1]-1]),Y_Pa-
int(M*HKy[1][HNy[1][i-1]-1]));
 Image1->Canvas->LineTo(X_Pa+int(M*HKy[0][HNy[3][i-1]-1]),Y_Pa-
int(M*HKy[1][HNy[3][i-1]-1]));
 Image1->Canvas->LineTo(X_Pa+int(M*HKy[0][HNy[2][i-1]-1]),Y_Pa-
int(M*HKy[1][HNy[2][i-1]-1]));
 Image1->Canvas->LineTo(X_Pa+int(M*HKy[0][HNy[0][i-1]-1]),Y_Pa-
int(M*HKy[1][HNy[0][i-1]-1]));
 }
}
void __fastcall TForm2::Button2Click(TObject *Sender)
{
 if(Es1==0||Us1==0) RadioGroup1->ItemIndex=0;
switch(RadioGroup1->ItemIndex)
{
 case 0: Tugun_nuqta_koordinata(Image1->Canvas,HK,Us);break;
 case 1: Tugun_nuqta_koordinata(Image1->Canvas,HK1,Us1);break;
}
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button3Click(TObject *Sender)
```

```
{
Close();
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include<stdio.h>
#include "Unit3.h"
#include "Unit1.h"
#define  fayl   "umumiy.txt"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
int N;
mass1 M1,M2;
int Us,Es,*HN[4];
float *HK[2];
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm3::FormShow(TObject *Sender)
{
 Edit1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::BitBtn2Click(TObject *Sender)
{
 for(int i=0;i<N;i++) M1[i]=StrToInt(StringGrid1->Cells[i][0]);
 Mos_keltirish(M1,M2,N);
 Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::BitBtn1Click(TObject *Sender)
{
 N=StrToInt(Edit1->Text);
  StringGrid1->ColCount=N;
 FILE *out;
out=fopen(fayl,"r+");
fscanf(out,"%d",&Us);
fscanf(out,"%d",&Es);
for(int i=1;i<=2;i++)
HK[i-1]=new float[Us];
for(int i=1;i<=4;i++)
HN[i-1]=new int[Es];
```

```
for(int i=1;i<=Es;i++)
 for(int j=1;j<=4;j++)
  fscanf(out,"%d",&HN[j-1][i-1]);
for(int i=1;i<=Us;i++)
 for(int j=1;j<=2;j++)
  fscanf(out,"%f",&HK[j-1][i-1]);
fclose(out);

}
//---------------------------------------------------------------------------
void TForm3::Mos_keltirish(mass1 M1,mass1 M2,int N)
{
   int k=0,l=N,l1=N;
   bool p;
   for(int i=0;i<N;i++)
     M2[i]=i+1;

   while(1)
   {
     for(int i=k;i<l;i++)
     {
       for(int j=0;j<Es;j++)
       {
         for(int t=0;t<4;t++)
         {
           if(M1[i]==HN[t][j])
           {
             for(int t1=0;t1<4;t1++)
               if(t1!=t)
               {
                 p=true;
                 for(int j1=0;j1<l1;j1++)
                   if(M1[j1]==HN[t1][j])
                     p=false;

                 if(p)
                 {
                   M1[l1]=HN[t1][j];
                   l1++;
                 }
               }
           }
         }
       }

       for(int i=l;i<l1;i++)
         M2[i]=i+1;

    if(l1==Us)break;
    k=l;
    l=l1;
```

```
  }
int *HN1[4];
float *HK1[2];
for(int i=0;i<4;i++)
   HN1[i]=new int[Es];
for(int i=0;i<2;i++)
   HK1[i]=new float[Us];
for(int i=0;i<Es;i++)
 for(int j=0;j<4;j++)
  HN1[j][i]=HN[j][i];
for(int i=0;i<Us;i++)
 for(int j=0;j<2;j++)
  HK1[j][i]=HK[j][i];
/*for(int i=0;i<4;i++)
 HN1[i]=new int[Es];
for(int i=0;i<2;i++)
 HK1[i]=new float[Us]; */
for(int k=0;k<Us;k++){
for(int i=0;i<Es;i++)
{
 for(int j=0;j<4;j++)
  {
   if(HN[j][i]==M1[k])HN1[j][i]=M2[k];
  }
}
 float p1;
 p1=HK[0][M1[k]-1];
// HK1[0][M1[k]-1]=HK[0][M2[k]-1];
 HK1[0][M2[k]-1]=p1;
 p1=HK[1][M1[k]-1];
 //HK1[1][M1[k]-1]=HK[1][M2[k]-1];
 HK1[1][M2[k]-1]=p1;
}
int max,min,max_min=0,max_min1=0;
for(int i=0;i<Es;i++)
{
 max=HN[0][i];
 min=HN[0][i];
 for(int j=1;j<4;j++)
  {
   if(HN[j][i]>max) max=HN[j][i];
   if(HN[j][i]<min) min=HN[j][i];
  }
 if((max-min)>max_min)max_min=max-min;
}
for(int i=0;i<Es;i++)
{
 max=HN1[0][i];
 min=HN1[0][i];
 for(int j=1;j<4;j++)
  {
   if(HN1[j][i]>max) max=HN1[j][i];
```

```
   if(HN1[j][i]<min) min=HN1[j][i];
  }
 if((max-min)>max_min1)max_min1=max-min;
}
if(max_min1<max_min)
{
   StringGrid2->RowCount = Es;
   StringGrid2->ColCount = 5;

   FILE *f=fopen(fayl,"w+");
   fprintf(f,"%3d       %3d",Us,Es);
   fprintf(f,"\n");
   for(int i=1;i<=Es;i++)
   {
      for(int j=1;j<=4;j++)
      {
         fprintf(f,"%3d     ",HN1[j-1][i-1]);
         StringGrid2->Cells[j][i - 1] = HN1[j-1][i-1];
      }
      fprintf(f,"\n");
   }

   for(int i=1;i<=Us;i++)
   {
      for(int j=1;j<=2;j++)
        fprintf(f,"%3f    ",HK1[j-1][i-1]);
      fprintf(f,"\n");
   }

   fclose(f);
   ShowMessage("Длина ленты уменьшилась от "+IntToStr(2*(max_min+1)+1)+" до
"+IntToStr(2*(max_min1+1)+1));
}
}
```