Лабораторная работа № 4

Работа с двумерными матрицами

Цель работы: изучить методы и алгоритмы обработки информации, представленной в виде двумерных матриц.

Часть 1

Задание:

1. Сформировать матрицу с заданным количеством строк и столбцов и заполнить ее случайными числами в диапазоне от -100 до 100.
2. Построить модифицированную матрицу, заменив в исходной матрице все отрицательные элементы нулями.
3. Найти сумму элементов указанного столбца модифицированной матрицы.
4. Выполнить самостоятельные задания согласно варианту.

Подготовка к выполнению работы

Создаем новый проект и изменяем у формы свойство ***Caption*** на название лабораторной работы. Сохраняем проект в рабочую папку командой ***File → Save Project as***.

Формирование интерфейса программы

Для работы с двумерными матрицами удобно использовать компонент ***StringGrid*** со страницы ***Additional***. Это компонент, включает в себя двумерный массив текстовых строк, т.е. каждый его элемент имеет тип ***string***. Основные свойства компонента ***StringGrid*** представлены в таблице:

Свойство	Назначение
ColCount	Количество столбцов.
RowCount	Количество строк.
Col	Номер текущего столбца.
Row	Номер текущей строки.
Cols[NCol]	Одномерный массив ячеек таблицы указанного столбца.
Rows[NRow]	Одномерный массив ячеек таблицы указанной строки.
Cells[NCol, NRow]	Двумерный массив ячеек таблицы (текстовых строк). При обращении в некоторой ячейке первым указывается номер столбца ***NCol***, вторым – номер строки ***NRow***. Как и во всех компонентах, имеющих свойства-списки, номера столбцов и строк отсчитываются от 0.
FixedCols	Количество зафиксированных столбцов. Зафиксированные столбцы выделяются цветом, и в них

	нельзя вводить данные. Как правила, ячейки зафиксированных столбцов используются в качестве заголовков таблиц.
FixedRows	Количество зафиксированных строк.
Options.goEditing	Признак допустимости редактирования содержимого ячеек (*true* – редактирование разрешено, *false* – запрещено).

Для группирования элементов интерфейса программы будем использовать компонент-контейнер ***GroupBox***.

Для создания удобного и наглядного интерфейса программы рассмотрим свойство ***Align*** (выравнивание), присутствующее у многих компонентов, в том числе у ***GroupBox*** и ***StringGrid***. Оно позволяет управлять расположением компонента на компоненте-контейнере и может принимать следующие значения:

- *alNone* – значение по умолчанию. Компонент будет расположен там, где его разместит программист, а размеры компонента будут устанавливаться независимо.
- *alButtom* – компонент «прижмётся» к нижнему краю контейнера и растянется по всей его ширине. Таким образом, мы не сможем непосредственно управлять шириной компонента как таковой, поскольку она всегда будет следовать за шириной контейнера.
- *alTop* – компонент «прижмётся» к верхней границе контейнера и растянется по всей его ширине.
- *alLeft* – компонент «прижмётся» к левому краю контейнера и растянется по всей его высоте.
- *alRight* – компонент «прижмётся» к правому краю контейнера и растянется по всей его высоте.
- *alClient* – компонент займёт всё доступное пространство контейнера.

Разместим на форме следующие компоненты:

- ***GroupBox*** – для исходной таблицы. В свойстве ***Caption*** укажем строку *Исходная матрица*. Свойству ***Align*** присвоим значение *alTop***.** Поскольку для этого компонента контейнером является форма, он «прижмется» к ее верхнему краю. Поэкспериментируйте с изменением ширины компонента и ширины формы и подберите подходящий вариант.
- ***StringGrid*** – для формирования исходной матрицы (***Name*** = *'StringGridFirst'*). Разместим таблицу на компоненте ***GroupBox*** и свойству ***Align*** присвоим значение *alClient***.** Обнулим у нее количество фиксированных строк и столбцов (рис. 4.1).

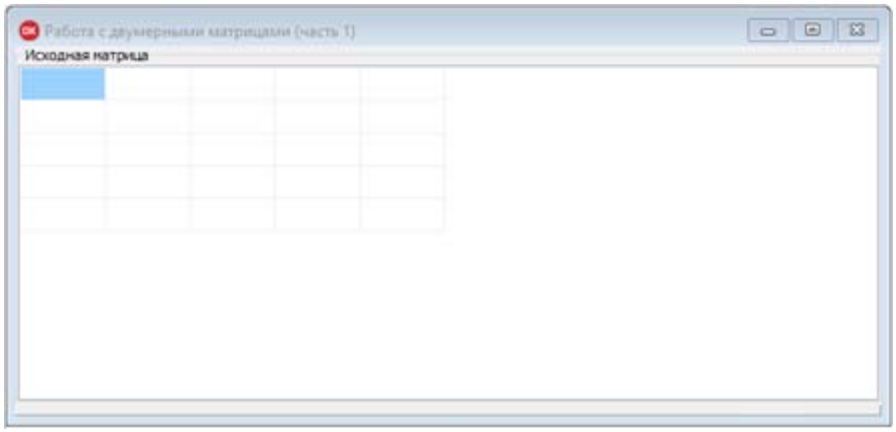

Рис. 4.1. Форма с исходной таблицей

Разместим компонент ***StringGrid*** для модифицированной матрицы, как показано на рис. 4.2 (***Name*** = *'StringGridModif'*).

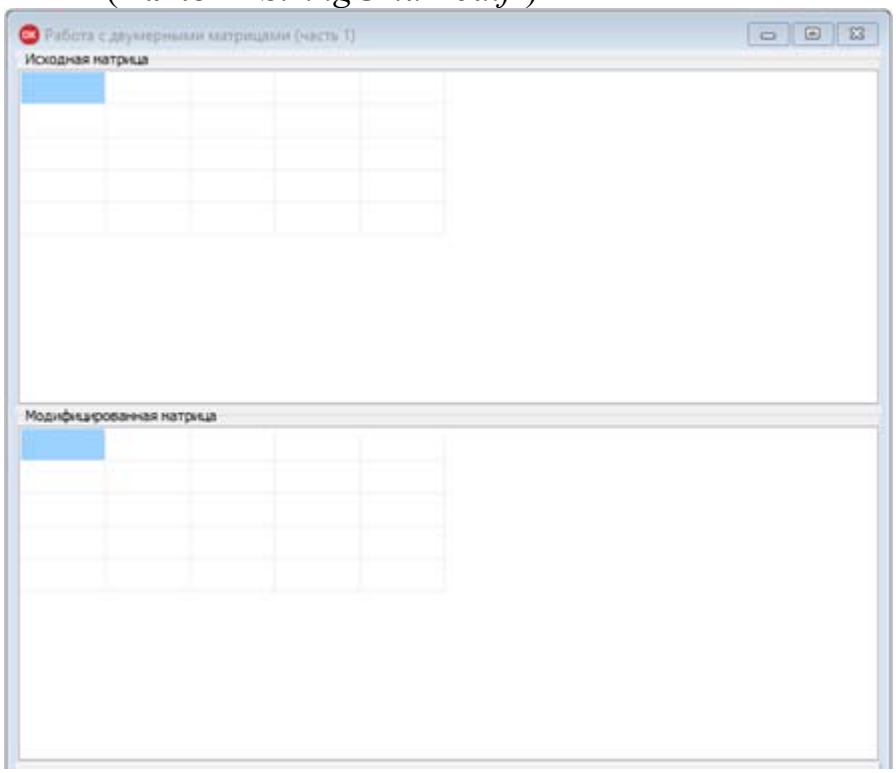

Рис. 4.2. Форма с двумя таблицами

Для группирования элементов управления воспользуемся компонентом-контейнером ***Panel***, который разместим его ниже обеих таблиц (***Caption*** = ''; ***Align*** = *alClient*)**.**

На контейнере ***Panel*** разместим следующие компоненты:

- Два редактора ***Edit*** (***Name*** = '*EditCol* ' и ***Name*** = '*EditRow* ') для ввода количества столбцов и строк. Свойству ***Text*** присвоим значение пустой строки.
- Две метки ***Label*** для описания назначения редакторов ***Edit***.
- Кнопку ***Button*** для формирования исходной матрицы (***Caption*** = '*Матрица*', ***Name*** = '*ButtonFirst* ').

- Кнопку ***Button*** для формирования модифицированной матрицы (***Caption*** = ’*Модифицированная матрица*’, ***Name*** = ’*ButtonModif* ’). Поскольку до формирования исходной матрицы формировать модифицированную матрицу не имеет смысла, сделаем этот и все последующие компоненты невидимыми. Для этого их свойствам ***Visible*** присвоим значение *false*. Сразу после формирования исходной матрицы сделаем компонент видимым.
- Редактор ***Edit*** для ввода номера столбца, в котором будут суммироваться элементы (***Visible*** = *false,* ***Name*** = ’*EditNumCol* ’).
- Метку ***Label*** для описания назначения этого редактора (***Visible*** = *false,* ***Name*** = ’*LabelNumCol* ’).
- Метку ***Label*** для вывода суммы (***Visible*** = *false,* ***Name*** = ’*LabelSum*’).
- Кнопку Button для расчета суммы (***Visible*** = *false,* ***Name*** = ’*ButtonSum*’).

Конструирование интерфейса закончено. Окончательный вид окна программы приведен на рис. 4.3.

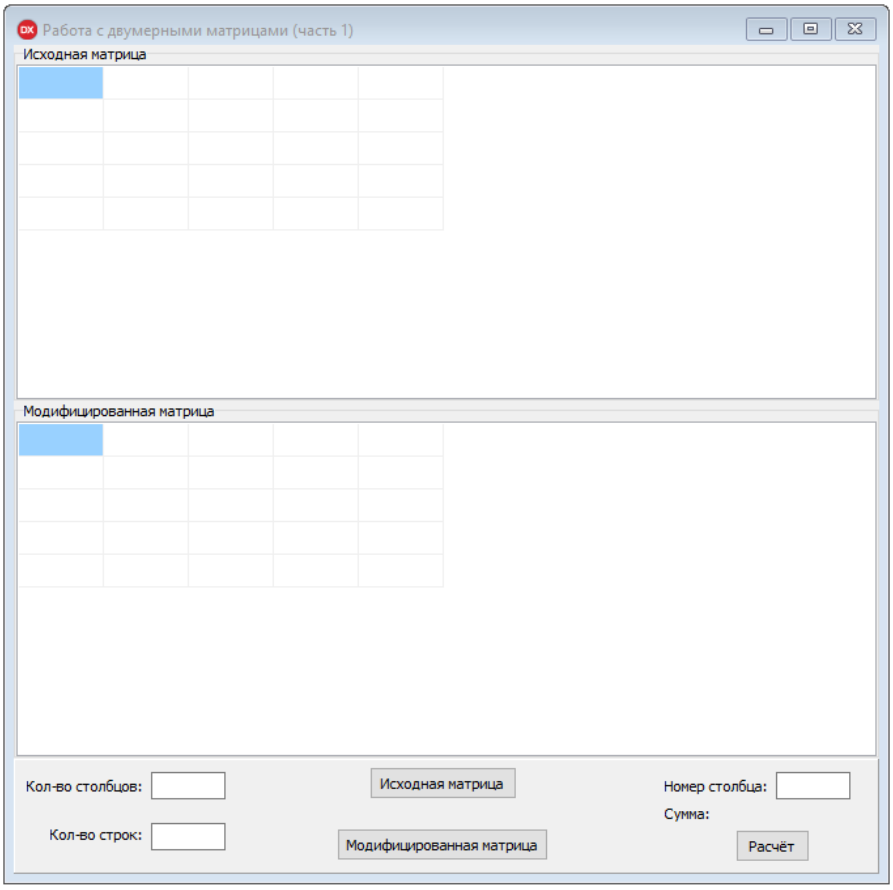

Рис. 4.3. Интерфейс программы

---

*Как можно заметить, компоненты перемещаются по форме не плавно, а сразу на несколько условных единиц. Для формирования красивого интерфейса это может создать определённые трудности. Данную функцию можно отключить в опциях (**Tools → Options…**) группы **Form Designer**, сняв флажок с пункта **Snap to grid**.*

*Изменять расположение компонента на форме можно с помощью свойств **Top** и **Left** этого компонента, которые отвечают за удалённость левого верхнего края компонента от верхней и левой границ контейнера соответственно*

Формирование исходной матрицы

Процедура формирования исходной матрицы представляет собой обработчик события ***OnClick*** кнопки ***ButtonFirst***. Перед началом построения необходимо проверить корректность исходных данных. Прежде всего, выясним, введены ли какие-либо данные в компоненты ***Edit***. Пример такой проверки для ***Edit_Col*** имеет вид:

```
if Edit_Col.Text = '' then
  begin
    ShowMessage('Введите количество столбцов.');
    Edit_Col.SetFocus;
    Exit;
          end;
```

*Аналогичную проверку сделайте для компонента **Edit_Row**.*

Для построения исходной матрицы необходимо определить количество столбцов и строк в ней, которые были введены в компонентах ***EditCol*** и ***EditRow*** соответственно:

```
StringGridFirst.ColCount:= StrToInt(EditCol.Text);
StringGridFirst.RowCount:= StrToInt(EditRow.Text);
```

Аналогично тому, как это делалось в предыдущих лабораторных работах, запретите вводить в компоненты ***EditCol*** и ***EditRow*** любые значения, кроме цифр и клавиши удаления символа**.**

Для заполнения матрицы случайными числами воспользуемся датчиком случайных чисел – процедурой ***Random(max)***, который генерирует случайное число в диапазоне от *0* до *max*. Для смещения диапазона в интервал от *-100* до *100* зададим значение ***max*** равное *200* и вычтем полученное число из константы *100*. Перед использованием процедуры ***Random*** ее необходимо инициализировать процедурой ***Randomize***.

Для заполнения всей таблицы организуем два вложенных цикла **for**, «пробегающих» по ее столбцам и строкам, не забыв добавить описания переменных цикла (***var i, j: Integer;***):

```
Randomize;
for i := 0 to StringGridFirst.ColCount - 1 do
for j := 0 to StringGridFirst.RowCount - 1 do
StringGridFirst.Cells[i, j]:= IntToStr(100 - Random(200));
```

Поскольку нумерация столбцов и строк начинается с нуля, максимальные значения переменных циклов необходимо уменьшить на единицу.

Последнее, что необходимо в этом обработчике, – сделать кнопку ***ButtonModif*** видимой:

ButtonModif.Visible:= true;

Запускаем полученную программы, вводим количество столбцов и строк, нажимаем кнопку *Матрица* и получаем результат (рис. 4..4).

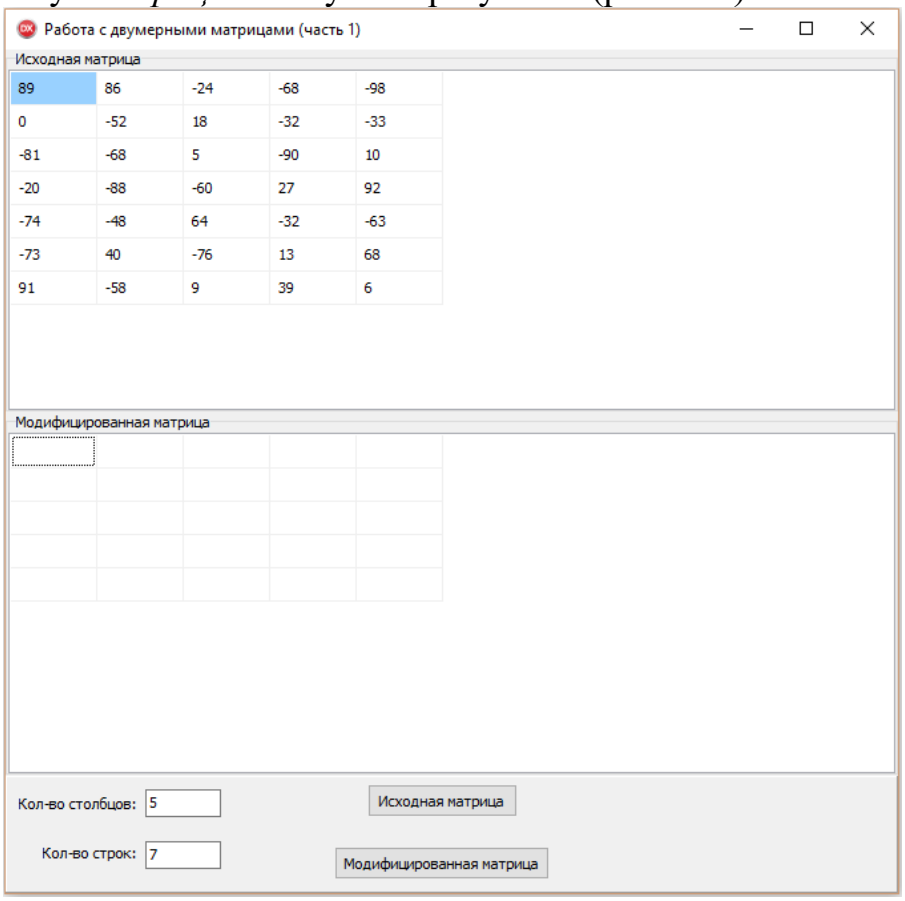

Рис. 4.4. Исходная матрица

Формирование модифицированной матрицы

Для формирования модифицированной матрицы необходимо, прежде всего, сделать ее такой же, как и исходная матрица, по количеству столбцов и строк. Это делается в обработчике события ***OnClick*** кнопки ***ButtonModif***::

```
procedure TForm1.ButtonModifClick(Sender: TObject);
begin
 StringGridModif.ColCount:= StringGridFirst.ColCount;
 StringGridModif.RowCount:= StringGridFirst.RowCount;
end;
```

По заданию модифицированная матрица должна копировать исходную матрицу, только отрицательные элементы должны обращаться в ноль. Для этого снова воспользуемся двойным циклом. Организуем два цикла ***for***, «пробегающих» по столбцам и строкам таблицы ***StringGridFirst*** и сравним каждый элемент с нулем: положительный элемент запишем в модифицированную матрицу без изменений, а отрицательный – заменим нулем:

```
for i := 0 to StringGridFirst.ColCount - 1 do
for j := 0 to StringGridFirst.RowCount - 1 do
  if StrToInt(StringGridFirst.Cells[i, j]) < 0 then
    StringGridModif.Cells[i, j]:= '0'
 else
  StringGridModif.Cells[i, j]:= StringGridFirst.Cells[i, j];
```

Делаем видимыми компоненты, необходимые для расчета суммы:

```
LabelNumCol.Visible:= true;
EditNumCol.Visible:= true;
LabelSum.Visible:= true;
ButtonSum.Visible:= true;
```

Запускаем программу и проверяем результат (рис. 4.5).

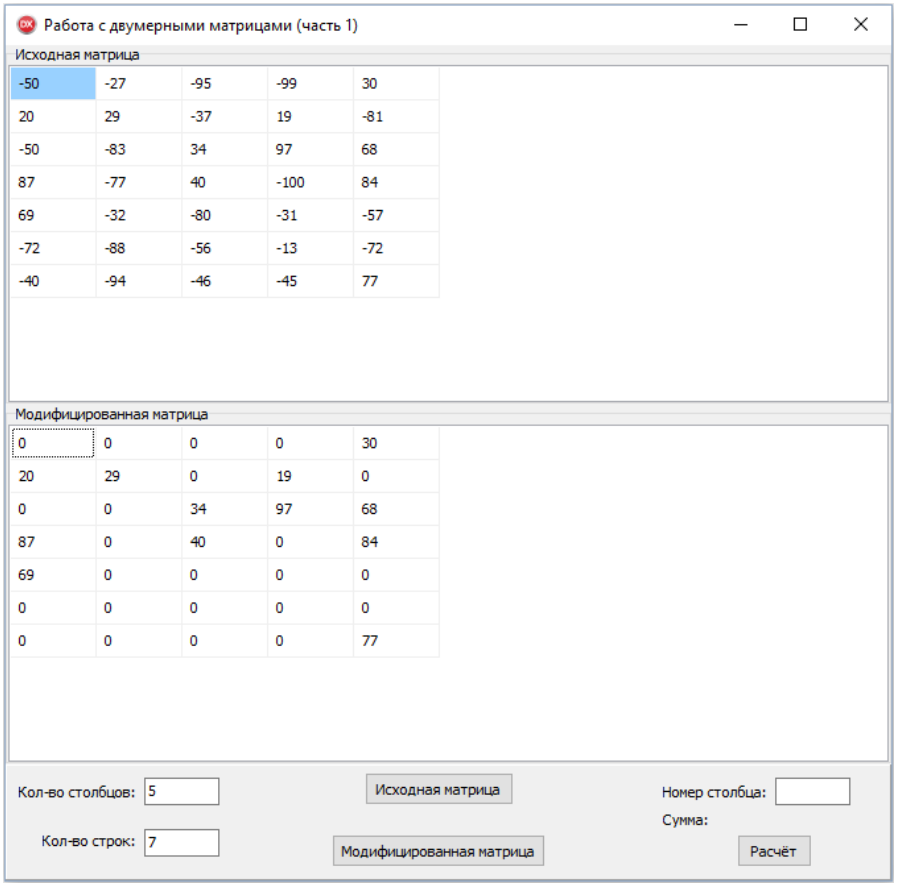

Рис. 4.5. Исходная и модифицированная матрицы

Расчет суммы

Переходим к последнему этапу – расчёту суммы. Создаём обработчик события ***OnClick*** компонента ***Button_Sum***. Первым делом определим переменную для обозначения номера столбца, элементы которого нужно просуммировать, и присвоим ей введенное значение:

NumCol := StrToInt(Edit_NumCol.Text) – 1;

*Значение, содержавшееся в поле ввода **Edit_NumCol**, необходимо уменьшить на единицу, потому что обычно человек начинает счёт с единицы, а столбцы компонента **StringGrid**, как уже было сказано выше, нумеруются с нуля.*

Введём ещё одну проверку: не превышает ли введённый номер столбца количества столбцов в таблице ***StringGrid_Modif***:

```
if ncol > StringGridModif.ColCount - 1 then
  begin
    ShowMessage('Неверный номер столбца');
    exit;
  end;
```

Далее организовываем цикл for по всем строкам указанного столбца. Перед началом цикла переменной sum, в которой будет накапливаться сумма, надо обязательно присвоить нулевое значение.

```
sum:= 0;
for i := 0 to StringGridModif.RowCount - 1 do
sum:= sum + StrToInt(StringGridModif.Cells[ncol, i]);

LabelSum.Caption:= 'Сумма = ' + IntToStr(sum);
```

*1. Организуйте проверку корректности исходных данных, вводимых в компоненте **Edit_NumCol**.*

*2. Для каждого поля ввода сделайте ограничение на ввод символов, как это было в предыдущих работах (обработчик события **OnKeyPress**). Не забудьте, что в данной программе нужно разрешить ввод только целых положительных чисел.*
3. Попробуйте ввести в поле ввода номера столбца ноль. Дополните программу так, чтобы пользователю выводилось сообщение о том, что ввод нуля в данное поле (а если подумать, то и в остальные тоже) недопустим.

4. Сделайте так, чтобы при ошибках ввода данных программа реагировала адекватно (например, в случае ошибки становились невидимыми те или иные компоненты, происходила очистка матриц и т.п.)

Варианты заданий для самостоятельной работы

Задание:

1. Сформировать матрицу с заданным количеством строк и столбцов и заполнить её случайными числами в диапазоне от -100 до 100.
2. Построить модифицированную матрицу по правилу, указанному в первом столбце соответствующего варианта.
3. Произвести расчёты (обязательно по нажатию кнопки), указанные в втором столбце соответствующего варианта..

1. Задания, в которых фигурирует главная или побочная диагональ, работают только с квадратными матрицами. В этих случаях необходимо вводить только одно число, которое будет определять и количество столбцов, и количество строк.

2. В заданиях, в которых сказано о половинах матрицы, при нечетном количестве строк или столбцов среднюю строку или столбец оставлять на том же месте в модифицированной матрице.

Вариант	**Задание на формирование модифицированной матрицы**	**Задание на расчет по модифицированной матрице**
1	Построить верхнюю треугольную матрицу. Главную диагональ и все элементы, расположенные выше нее, оставить без изменения, а остальные заменить нулями.	Найти сумму нечетных элементов указанного столбца
2	Поменять местами левую и правую половины матрицы	Найти сумму тех элементов указанной строки, которые меньше своих соседей справа. Последний элемент включить в сумму
3	Построить симметричную матрицу относительно главной диагонали	Найти максимальный и минимальный элементы главной диагонали
4	Построить матрицу по правилу: $m_{ij} = f_{ij} + i - 2j$, где m_{ij} – элемент	Найти сумму четных элементов указанной строки

	модифицированной матрицы, f_{ij} – элемент исходной матрицы	
5	Расположить все элементы каждой строки матрицы в обратном порядке	Найти количество положительных элементов, стоящих выше главной диагонали.
6	Во всех строках матрицы сдвинуть элементы на одну позицию вправо. Последний элемент поместить на место первого элемента.	Найти сумму тех элементов указанного столбца, которые меньше своих соседей снизу. Последний элемент включить в сумму
7	Построить нижнюю треугольную матрицу относительно побочной диагонали. Побочную диагональ и все элементы, расположенные ниже нее, оставить без изменения, а остальные заменить нулями	Ввести число и найти алгебраическую сумму элементов матрицы по правилу: если элемент меньше введенного числа, брать его со знаком минус, иначе – со знаком плюс
8	Построить симметричную матрицу относительно ее середины по вертикали	Ввести два числа и найти количество элементов матрицы из интервала, образованными этими числами
9	Построить «диагональный крест». Элементы главной и побочной диагоналей оставить без изменения, а остальные заменить нулями	Найти сумму элементов главной и побочной диагоналей
10	Для всех строк поменять местами элементы, стоящие в столбцах с нечетными номерами, с элементами, стоящими в столбцах с четными номерами	Ввести два числа и найти алгебраическую сумму элементов матрицы по правилу: если элемент попадает в интервал, образованный этими числами, брать его со знаком минус, иначе – со знаком плюс
11	В каждом столбце модифицированной матрицы элемент представляет собой сумму соответствующего элемента исходной матрицы и элемента из столбца, расположенного правее. Последний столбец модифицированной матрицы совпадает с последним столбцом исходной матрицы.	Найти суммы элементов матрицы, больших введенного числа
12	Во всех столбцах матрицы	Ввести два числа и найти

	сдвинуть элементы на одну позицию вниз. Последний элемент поместить на место первого элемента.	количество элементов матрицы, не входящих в интервал, образованный этими числами
13	Транспонировать матрицу, т.е. поменять местами строки и столбцы	Найти сумму элементов указанного столбца, находящихся в строках с четными номерами
14	Записать в обратном порядке элементы всех столбцов матрицы	Найти максимальный и минимальный элементы побочной диагоналей
15	Построить симметричную матрицу относительно побочной диагонали	Найти количество положительных элементов, стоящих выше побочной диагонали
16	Первый и последний столбцы модифицированной матрицы совпадают со столбцами исходной матрицы. В остальных столбцах каждый элемент равен сумме двух соседних по строке элементов исходной матрицы.	Найти максимальный и минимальный элементы указанного столбца
17	Построить логическую матрицу по следующему правилу: все положительные элементы заменить единицами, а отрицательные – нулями	Найти количество нулей и единиц в матрице
18	В каждой строке модифицированной матрицы элемент представляет собой сумму соответствующего элемента исходной матрицы и элемента из строки, расположенной ниже. Последняя строка модифицированной матрицы совпадает с последней строкой исходной матрицы.	Ввести два числа и найти количество элементов из интервала, образованного этими числами, не считая элементы главной диагонали
19	Построить симметричную матрицу относительно ее середины по горизонтали	Найти алгебраическую сумму элементов указанной строки по правилу: если второй индекс меньше первого индекса, брать его со знаком минус, иначе – со знаком плюс
20	Поменять местами верхнюю и нижнюю половины матрицы	Взять первый и последний элементы матрицы и определить максимум и минимум из них.

		Найти количество элементов матрицы, больших минимума и меньших максимума
21	Первая строка модифицированной матрицы совпадает с первой строкой исходной матрицы. Далее в каждой строке модифицированной матрицы элемент представляет собой сумму соответствующего элемента исходной матрицы и элемента из строки, расположенной выше.	Найти количество четных и нечетных элементов указанного столбца
22	Для всех строк поменять местами элементы, стоящие в столбцах с нечетными номерами, с элементами, стоящие в столбцах с четными номерами	Найти алгебраическую сумму элементов указанной строки по правилу: если сумма индексов элемента четная, брать его со знаком плюс, иначе – со знаком минус
23	Первый и последний столбец модифицированной матрицы совпадают со столбцами исходной матрицы. В остальных столбцах каждый элемент равен единице, если он больше обоих соседних по строке элементов исходной матрицы, и нулю в противном случае.	Ввести два числа и найти количество элементов матрицы из интервала, образованного этими числами
24	Построить нижнюю треугольную матрицу. Главную диагональ и все элементы, расположенные ниже нее, оставить без изменения, а остальные заменить нулями	Найти максимальный и минимальный элементы указанной строки
25	Крайние строки и столбцы модифицированной матрицы заполнить нулями, а остальные элементы перенести без изменения.	Найти сумму тех элементов указанной строки, которые больше своих соседей слева. Первый элемент не включить в сумму
26	Записать в обратном порядке элементы главной диагонали матрицы	Найти сумму элементов указанной строки, находящихся в столбцах с нечетными номерами
27	Первый столбец	Найти алгебраическую сумму

	модифицированной матрицы совпадает с первым столбцом исходной матрицы. Далее в каждом столбце модифицированной матрицы элемент представляет собой сумму соответствующего элемента исходной матрицы и элемента из столбца, расположенного левее.	элементов указанного столбца по правилу: если первый индекс элемента больше второго, брать его со знаком плюс, иначе – со знаком минус
28	Первая и последняя строки модифицированной матрицы совпадают со строками исходной матрицы. В остальных строках каждый элемент равен единице, если он меньше обоих соседних по столбцу элементов исходной матрицы, и нулю в противном случае.	Найти количество четных и нечетных элементов указанной строки
29	Построить верхнюю треугольную матрицу относительно побочной диагонали. Побочную диагональ и все элементы, расположенные выше нее, оставить без изменения, а остальные заменить нулями	Найти алгебраическую сумму элементов указанного столбца по правилу: если сумма индексов элемента нечетная, брать его со знаком плюс, иначе – со знаком минус
30	Первая и последняя строки модифицированной матрицы совпадают со строками исходной матрицы. В остальных строках каждый элемент равен сумме двух соседних по столбцу элементов исходной матрицы.	Найти максимальный элемент главной и побочной диагоналей

Часть 2

Задание:

1. Сформировать матрицу, количество строк и столбцов которой случайное число в интервале от 2 до 10, и заполнить ее элементами ряда $\frac{6}{n^2+5n+6}$.

2. Указать мышью строку и вывести все ее элементы, сложенные с элементами первой строки (если щелкаем левой кнопкой мыши) и сложенные с

элементами последней строки (если щелкаем правой кнопкой мыши). Во всех остальных случаях никаких действий не выполняется.

3. Выполнить самостоятельные задания согласно варианту.

Подготовка к выполнению работы

Создаем новый проект и изменяем у формы свойство ***Caption*** на название лабораторной работы. Сохраняем проект в рабочую папку командой ***File → Save Project as***.

Формирование интерфейса программы

Для выполнения поставленной задачи необходимо вывести таблицу, а также строку, которая будет результатом сложения элементов указанной строки с первой или последней строкой.

Создаем интерфейс программы, показанный на рис. 4.6. Для того чтобы при любых значения количества столбцов и строк таблица была видна на экране полностью, зададим максимальные значения – *10*. Исходя из этого, изменяем размеры формы и компонента ***GroupBox***.

Основные свойства компонентов:

- ***StringGrid*** для формирования исходной матрицы: ***Name*** = ’*StringGridFirst.*’
- ***StringGrid*** для строки: ***Name*** = ’*StringGridStr* ’.
- ***Button*** для формирования исходной матрицы: ***Caption*** = ’*Матрица*’, ***Name*** = ’*ButtonFirst* ’.

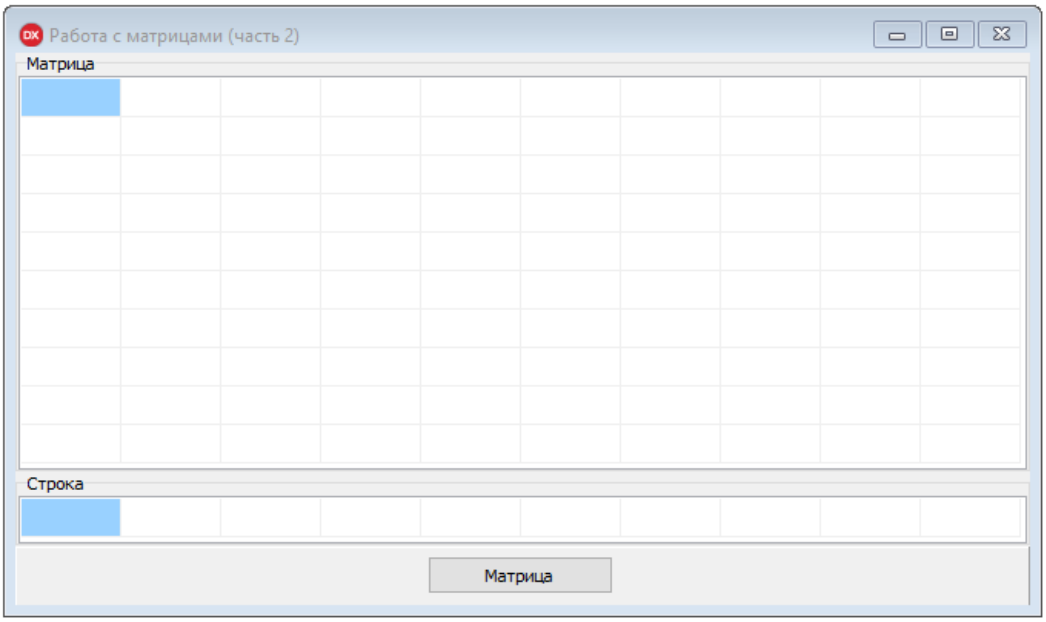

Рис. 4.6. Интерфейс программы

Формирование матрицы

Создаем обработчик события ***OnClick*** кнопки ***Button_First***. Построение матрицы начинается с определения (по заданию – случайного) количества её столбцов и строк. Для этого воспользуемся функцией ***Random*** (не забыв перед этим написать процедуру ***Randomize***):

```
procedure TForm1.ButtonFirstClick(Sender: TObject);
var i, j: integer;
begin
 Randomize;
 StringGridFirst.ColCount:= 2+ Random(8);
 StringGridFirst.RowCount:= 2+ Random(8);
end;
```

Для расчета элементов ряды введем переменную ***n*** – номер элемента (описать в секции ***var***). Первоначальное значение ***n***=*1*. Для прохода по всем элементам таблицы используем вложенный цикл ***for***. Обратите внимание, что тело цикла состоит из двух команд: присвоения значения ячейке таблицы и увеличения значения счетчика ***n***. Их необходимо объединить в один составной оператор операторными скобками ***begin end***.

Поскольку элементы ряда являются действительными числами, для их перекодировки в текстовые строки используем функцию ***FloatToStr***. Функция ***sqr*** возвращает квадрат числа.

```
 n:= 1;
 for i := 0 to StringGridFirst.ColCount - 1 do
 for j := 0 to StringGridFirst.RowCount - 1 do
 begin
 StringGridFirst.Cells[i, j]:= FloatToStr(6 / (sqr(n) + 5*n + 6));
 Inc(n);
 end;
```

Последней командой сделаем количество столбцов в строке таким же, как в исходной матрице:

```
StringGrid_Str.ColCount := StringGrid_First.ColCount;
```

Заполнение строки

Для заполнения строки необходимо выбрать щелчком мыши некоторую строку исходной матрицы. Другими словами, строка должна заполниться в соответствии с заданием в момент щелчка мышью в исходную матрицу. Это означает, что необходимо написать обработчик события ***OnMouseDown*** таблицы ***StringGridFirst***.

procedure TForm1.StringGridFirstMouseDown(Sender: TObject;
Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin

end;

Создав данный обработчик, обратите внимание на параметры процедуры. Параметр ***Button*** имеет тип:

TMouseButton = (*mbLeft, mbRight, mbMiddle*);
где *mbLeft* – была нажата левая клавиша мыши;
mbRight – была нажата правая клавиша мыши;
mbMiddle – была нажата средняя клавиша мыши.

Параметр ***Shift*** имеет тип:
TShiftState = **set of** (*ssShift, ssAlt, ssCtrl, ssLeft, ssRight, ssMiddle, ssDouble*);
т.е. представляет собой множество, содержащее одно или несколько следующих значений:

- *ssShift* – нажата клавиша Shift;
- *ssAlt* – нажата клавиша Alt;
- *ssCtrl* – нажата клавиша Ctrl;
- *ssLeft* – нажата левая клавиша мыши;
- *ssRight* – нажата правая клавиша мыши;
- *ssMiddle* – нажата средняя клавиша мыши;
- *ssDouble* – нажаты и правая и левая клавиши мыши.

Обратим внимание, что и параметр ***Button*** и параметр ***Shift*** имеют одинаковые значения, связанные с кнопками мыши (*..Left*, *..Right*, *..Middle*). Однако в этих параметрах они имеют различный смысл. В параметре ***Button*** передаётся код клавиши, нажатой именно в данный момент, а в параметре ***Shift*** содержатся коды клавиш, нажатых и ранее. Например, если вначале была нажата левая кнопка мыши, а затем правая, то код правой клавиши передастся в параметре ***Button*** (***Button*** = *mbRight*), а в параметре ***Shift*** передастся и код правой, и код левой клавиши: ***Shift*** = [*ssLeft, ssRight*].

Параметры ***X*** и ***Y*** – это координаты курсора в момент нажатия клавиши мыши, отсчитываемые от левой верхней точки таблицы с координатами (*StringGridFirst.Left*, *StringGridFirst.Top*). Таким образом, левый верхний угол таблицы имеет координаты (0, 0).

При работе с таблицами чаще требуются не координаты точки, в которой произошел щелчок мыши, в пикселах, а номер столбца и строки для той ячейки, в которой находился курсор мыши в момент нажатия. Определить эти значения поможет функция:

StringGrid.MouseToCell(X, Y, ACol, ARow);

где ***ACol*** – номер столбца, ***ARow*** – номер строки ячейки, в которой находился курсор в момент нажатия клавиши мыши.

Определим следующие локальные переменные:

- ***nc*** – номер столбца, в котором произошел щелчок;
- ***nr*** – номер строки, в которой произошел щелчок;
- ***r*** – переменная для хранения текущего значения ячейки (действительный тип).

Прежде всего, вычислим номер столбца и номер строки выбранной ячейки:

```
StringGridFirst.MouseToCell(X, Y, nc, nr);
```

Далее необходимо проверить, находится ли курсор мыши внутри таблицы в момент щелчка. Это означает, что значения ***nc*** и ***nr*** должны быть положительными и не превышать номера последнего столбца и последней строки:

```
.
if (nc > StringGridFirst.ColCount - 1) or (nr > StringGridFirst.RowCount - 1) or
   (nc < 0) or (nr < 0) then
 begin
   ShowMessage('Неверное указание ячейки');
   exit;
 end;
```

Заполняем строку – проходим по всем столбцам исходной матрицы циклом ***for***. В теле цикла необходимо сделать следующее:

- получить очередное значение ячейки таблицы в переменную ***r***;
- если была нажата левая кнопка мыши, то сложить его с соответствующей ячейкой нулевой строки, а если правая – то с соответствующей ячейкой последней строки, номер которой ***StringGridFirst.RowCount – 1***;
- записать полученное значение в нужную ячейку строки.

Поскольку в теле цикла несколько команд, все их надо заключить в операторные скобки ***begin end***.

```
for i:= 0 to StringGridFirst.ColCount - 1 do
 begin
   r:= StrToFloat(StringGridFirst.Cells[i, nr]);

   if Button = mbLeft then
     r:= r + StrToFloat(StringGridFirst.Cells[i, 0])
   else
    if Button = mbRigth then
   r:= r + StrToFloat(StringGridFirst.Cells[i, StringGridFirst.RowCount - 1]);

   StringGridStr.Cells[i, 0]:= FloatToStr(r);
end;
```

Запускаем программу и проверяем ее работу (рис. 4.7).

Организуйте очистку строки при формировании новой матрицы.

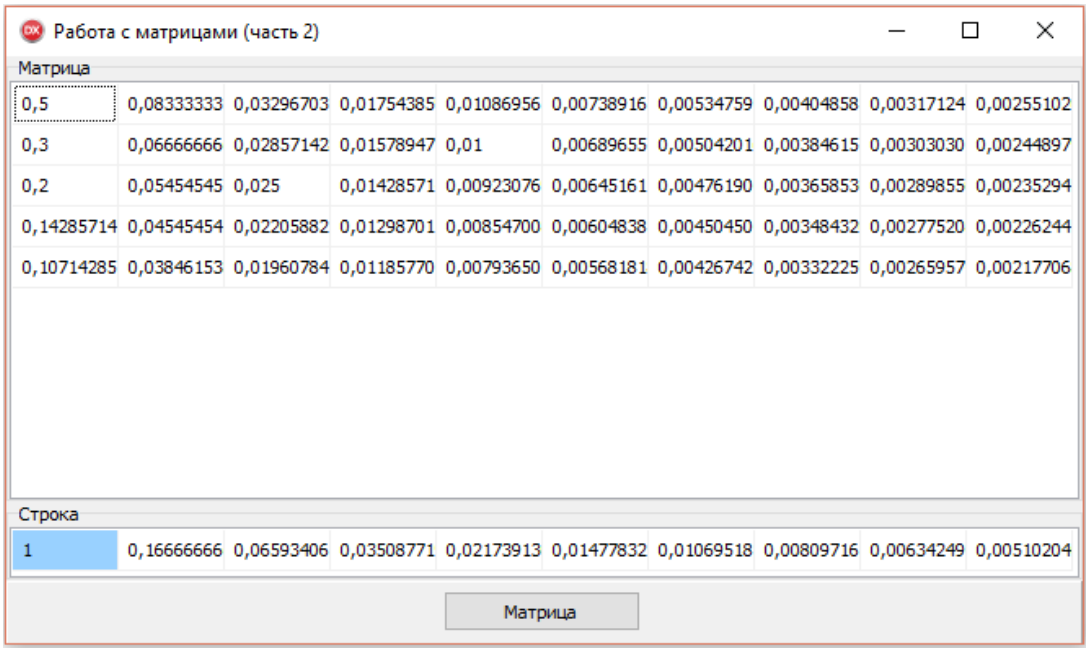

Рис. 9. Результат работы программы

Варианты заданий для самостоятельной работы

Задание:

1. Сформировать матрицу, количество строк и столбцов которой случайное число в интервале от 2 до 10, и заполнить её элементами ряда, указанного в варианте.
2. Вывести указанные элементы матрицы в строку по правилам, указанным в варианте в двух крайних правых столбцах.

*1. Для возведения в степень используется функции **power(x, n: real): real**; где **x** – число, которое надо возвести в степень **n**. Функция находится в модуле **Math**. Его следует добавить в описание **Uses**. Данную функцию запрещается использовать в знакопеременных рядах для определения знаков элементов ряда!*

*2. Для возведения экспоненты в степень используется функция **exp(x:real): real;***

*3. Для нахождения натурального логарифма используется функция **ln(x:real): real;***

4. Для вычисления факториала надо самостоятельно написать функцию ***fact(n:integer): real****. Несмотря на то, что факториал по определению является целым числом, тип функции делаем вещественным. Это необходимо для того, чтобы избежать переполнения при вычислении значений факториала для больших чисел.*

5. Для нахождения целой части вещественного числа используется функция ***trunc(x:real): integer;***

6. Для нахождения дробной части вещественного числа используется функция ***frac(x:real): real.***

Варианты заданий для самостоятельной работы

Вариант	**Ряд**	**Задание**	
		при щелчке левой кнопкой	**при щелчке правой кнопкой**
1	$\left(\frac{2n^2+3}{n^2+4}\right)^n$	Вывести строку, на которой щелкнули.	Вывести столбец, на который щелкнули.
2	$\left(\frac{2}{n}-\frac{5}{n^2}+3\right)\cos\left(\frac{1}{2n}\right)$	Сложить элементы строки с элементами строки, расположенной ниже. Последнюю строку оставить без изменения.	Сложить элементы строки с элементами строки, расположенной выше. Первую строку оставить без изменения.
3	$\ln(1+\frac{1}{n})$	Вывести элементы строки через один, начиная с нулевого элемента	Вывести элементы столбца через один, начиная с нулевого элемента
4	$\frac{n^3}{(n+5)!}$	Разделить на 2 элементы строки	Разделить на 2 элементы столбца
5	$(1+n)^{\frac{n+1}{n}}$	Вывести элементы строки, которые при умножении на 10 имеют четную целую часть	Вывести элементы столбца, которые при умножении на 10 имеют четную целую часть
6	$\frac{n}{\sqrt[n]{n!}}$	Вывести ячейку	Вывести ячейку и все окружающие ее ячейки
7	$\frac{n!}{6^n}$	Вывести строку в обратном порядке	Вывести столбец в обратном порядке
8	$\frac{4^n}{n^4}$	Вывести строку и строку, расположенную ниже. Если указываем последнюю строку в матрице, то ее надо вывести	Вывести строку и строку, расположенную выше. Если указываем первую строку в матрице, то ее надо вывести дважды

		дважды	
9	$\ln\frac{n^3+6}{n^3+5}$	Вывести максимальный элемент строки	Вывести минимальный элемент строки
10	$\frac{15(n!)}{2^n(n^2+1)}$	Вывести целые части элементов строки	Вывести целые части элементов столбца
11	$\frac{n!}{5^n}\sin(\frac{1}{2^n})$	Из суммы элементов строки вычесть сумму элементов столбца	Из суммы элементов столбца вычесть сумму элементов строки
12	$\left(\frac{2n^3+3}{5n^2+4}\right)^n$	Вывести целые части элементов строки, расположенной ниже, или самой строки, если она последняя.	Вывести дробные части элементов строки, расположенной ниже, или самой строки, если она последняя.
13	$\frac{1}{(3n+2)\sqrt{\ln(n+3)}}$	Удвоить элементы строки	Утроить элементы строки
14	$\frac{1}{5^n+2}$	Вывести суммы пар соседних элементов строки. При нечетном количестве элементов последний элемент вывести без изменения	Вывести суммы пар соседних элементов столбца. При нечетном количестве элементов последний элемент вывести без изменения
15	$\frac{(n+1)!}{4^n}$	Вывести столбец и столбец, расположенный правее. Если указываем последний столбец в матрице, то его надо вывести дважды	Вывести столбец и столбец, расположенный левее. Если указываем первый столбец в матрице, то его надо вывести дважды
16	$\frac{1}{\sqrt{n+4}\cdot\ln(n^2+5n)}$	Вывести максимальный элемент столбца	Вывести максимальный элемент строки
17	$\left(\frac{2n}{n+1}\right)^n$	Вывести дробные части элементов строки	Вывести дробные части элементов столбца
18	$\left(1-\cos\frac{\pi}{n}\right)^n$	Сложить элементы строки с элементами строки, расположенной выше. Первую строку оставить без изменения.	Сложить элементы строки с элементами строки, расположенной ниже. Последнюю строку оставить без изменения.
19	$(-1)^{n+1}\frac{3^n}{(2n-1)^2}$	Вывести элементы строки с нечетной целой частью	Вывести элементы столбца с нечетной целой частью

20	$(-1)^n \frac{(2n)!}{5^n}$	Вывести строку и строку, расположенную ниже. Если указываем последнюю строку в матрице, то ее надо вывести дважды	Вывести столбец и столбец, расположенней правее. Если указываем последний столбец в матрице, то его надо вывести дважды
21	$\frac{\sqrt[n]{n}}{\sin(n^2)}$	Вывести ячейку и ячейку, расположенную правее, если она существует	Вывести ячейку и ячейку, расположенную левее, если она существует
22	$\frac{(n+3)!}{(n+1)}\ln(n)$	Вывести целые части элементов столбца расположенного правее, или самого столбца, если он последний.	Вывести дробные части элементов столбца расположенного правее, или самого столбца, если он последний.
23	$\frac{n!}{(n+1)!-n!}$	Вывести минимальный элемент столбца	Вывести минимальный элемент строки
24	$(-1)^{n+1} \frac{(3n+1)!}{e^n}$	Вывести целые части элементов строки, расположенной ниже или самой строки, если она последняя.	Вывести целые части элементов столбца, расположенного правее или самого столбца, если он последний
25	$\frac{(n+1)!}{e^n}\cos\left(\frac{3}{n+1}\right)$	Вывести разности пар соседних элементов столбца. При нечетном количестве элементов последний элемент вывести без изменения	Вывести разности пар соседних элементов строки. При нечетном количестве элементов последний элемент вывести без изменения
26	$\frac{4^n \sin(n)}{\ln(3n+1)}$	Удвоить элементы столбца	Утроить элементы столбца
27	$\frac{3^n}{n^3+2n^2+6n+3}$	Сложить элементы строки с элементом выбранной ячейки	Сложить элементы столбца с элементом выбранной ячейки
28	$\frac{\sin\left(\frac{n\pi}{n+1}\right)}{n^2+2n}$	Вывести максимальный элемент столбца	Вывести минимальный элемент столбца
29	$\frac{4!}{n^5+5n}\ln(2n+1)$	Вывести строку и строку, расположенную выше. Если указываем первую	Вывести столбец и столбец, расположенный левее. Если указываем первый

		строку в матрице, то ее надо вывести дважды	столбец в матрице, то его надо вывести дважды
30	$(-1)^{n+1}\left(\frac{n^2 \cdot \cos(3n+1)}{(2)}\right)^n$	Вывести дробные части элементов строки, расположен-ной ниже, или самой строки, если она последняя.	Вывести дробные части элементов столбца, расположенного правее, или самого столбца, если он последний

1) Какой компонент используется для группировки других различных компонентов?

2) Какое событие соответствует нажатию кнопки мыши на компоненте?

3) Что представляет собой тип TShiftState? Для чего он используется? Приведите примеры.

4) Перечислите команды осуществляющие: целочисленное деление, получение остатка от деления, отбрасывание дробной части, округление числа.