Министерство образования Республики Беларусь

Учреждение образования

«Гомельский государственный университет имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра вычислительной математики и программирования

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Курсовая работа

Вариант 23

Исполнитель:

студент группы ИТП-11 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Усманов С. Э.

Научный руководитель:

к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лубочкин А.В.

Гомель 2024

РЕФЕРАТ

Курсовая работа: 52 страниц, 21 рисунка, 9 источников.

**Ключевые слова**: одномерные массивы, операции над одномерными массивами, двумерные массивы, операции над двумерными массивами, методы сортировок, методы нахождения обратной матрицы, символьная информация, алгоритмы обработки символьной информации, системы линейных алгебраических уравнений, методы решения систем линейных алгебраических уравнений.

**Объект исследования**: среды TurboPascal, Delphi.

**Предмет исследования**: алгоритмы обработки одномерных массивов, алгоритмы обработки двумерных массивов, алгоритмы обработки символьной информации, алгоритмы решения систем линейных алгебраических уравнений.

**Цель курсовой работы**: изучение основных алгоритмов решения исследуемых задач, приобретение навыков программирования в средах TurboPascal, Delphi.

**Задачами курсовой работы** являются: изучение основных алгоритмов обработки одномерных массивов, изучение основных алгоритмов обработки двумерных массивов, изучение основных алгоритмов обработки символьной

информации, изучение основных алгоритмов решения систем линейных алгебраических уравнений, применение на практике полученных знаний.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](file:///C:\Users\angel\Downloads\Telegram%20Desktop\КУРСОВАЯ.docx#_Toc167921140)

[1 Обработка одномерных массивов 5](#_Toc168687532)

[2 Операции над двумерными массивами по заданным критериям 9](#_Toc168687533)

[3 Обработка символьной информации 15](#_Toc168687534)

[4 Методы решения систем линейных алгебраических уравнений 17](#_Toc168687535)

[5 Теоретическая часть 19](#_Toc168687536)

[ЗАКЛЮЧЕНИЕ 2](#_Toc168687537)2

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 2](#_Toc168687538)3

[ПРИЛОЖЕНИЕ А. Код программы для обработки одномерных массивов 2](#_Toc168687539)4

ПРИЛОЖЕНИЕ Б. [Код программы для обработки двумерных массивов 3](#_Toc168687540)2

[ПРИЛОЖЕНИЕ В. Код программы для обработки строк 4](#_Toc168687541)2

[ПРИЛОЖЕНИЕ Г. Код программы для обработки СЛАУ 44](#_Toc168687542)

[ПРИЛОЖЕНИЕ Д. Теоретическая часть 52](#_Toc168687543)

ВВЕДЕНИЕ

Язык программирования Pascal, разработанный профессором Николаусом Виртом в конце 1960-х годов, быстро трансформировался из академического инструмента в мощное средство для создания обширных программных проектов. Стандартизация языка в 1974 году способствовала его распространению и применению в различных областях программирования. Pascal отличается строгой типизацией, поддержкой структурного и процедурного программирования, а также возможностью работы с динамической памятью и созданием пользовательских типов данных.

История Pascal тесно связана с развитием вычислительной техники в 60-70 годах прошлого века, когда сложность вычислений на компьютерах требовала новых подходов к программированию. Pascal стал одним из первых языков, предложивших структурированный подход к написанию кода, что позволило упростить чтение, анализ и тестирование программ.

Эти аспекты Pascal делают его значимым языком в истории программирования и образовании, а также подчеркивают его влияние на последующие поколения языков и сред разработки.

Delphi, появившийся в 1993 году и основанный на Object Pascal, представляет собой интегрированную среду разработки, облегчающую процесс создания приложений благодаря визуальному подходу и мощным инструментам отладки. Delphi поддерживает объектно-ориентированное программирование, обработку исключений и перегрузку методов, что делает его идеальным выбором для разработки как простых, так и сложных корпоративных приложений. Современные версии Delphi продолжают расширять возможности разработчиков, включая поддержку многозвенной архитектуры и разработку приложений для различных платформ.

В курсовой работе рассматриваются основные алгоритмы обработки одномерных массивов, основные алгоритмы обработки двумерных массивов, основные алгоритмы обработки символьной информации, основные алгоритмы решения систем линейных алгебраических уравнений. Для этого были изучены следующие вопросы: основные операции для работы с одномерными массивами, такие как инициализация, сортировка, поиск и манипуляции с элементами массива; основы работы с матрицами, включая операции сложения, умножения, транспонирования, определения детерминанта, нахождения обратной матрицы; рассмотрение методов кодирования, поиска и замены подстрок в строках; изучение методов решения систем линейных алгебраических уравнений, таких как метод Гаусса, Крамера, блочный метод. Были разработаны приложения, реализующие решение поставленных задач на обработку одномерных массивов, двумерных массивов, строковой информации и решений систем линейных алгебраических уравнений.

# **Обработка одномерных массивов**

**Постановка задачи:** разработать приложение для осуществления над векторами следующих операций: сумма (содержит несколько подпунктов, см. в пример), нахождение максимального и минимального элементов вектора и их номеров, сортировка (в соответствии с вариантом), решение своей задачи (в соответствии с вариантом). При разработке приложения использовать следующие компоненты: MainMenu, OpenDialog, SaveDialog, UpDown. По нажатию соответствующих кнопок должны происходить запись и чтение из файла вектора и его размерности.

**Компоненты страницы Dialog**

В состав Windows входит ряд стандартных типовых диалоговых окон, таких как окно выбора загружаемого файла, окно выбора шрифта и т.д. Работа со стандартными диалоговыми окнами осуществляется в три этапа.

Вначале на форму помещается соответствующий компонент и осуществляется настройка его свойств.

На втором этапе осуществляется вызов стандартного для диалогов метода Execute, который создает и показывает настроенное окно на экране. Вызов этого метода обычно располагается внутри обработчика какого-либо события. Например, обработчик выбора опции меню Открыть файл может вызвать метод Execute диалога ОpenDialog, обработчик нажатия кнопки Сохранить может вызвать такой же метод у компоненты SaveDialog. Только после обращения к Execute на экране появляется соответствующее диалоговое окно. Окно диалога является модальным окном, поэтому сразу после обращения к Execute дальнейшее выполнение программы приостанавливается до тех пор, пока пользователь не закроет окно. Функция Execute возвращает значение True, если результат диалога с пользователем был успешным.

Проанализировав результат Execute, программа может выполнить третий этап – использование введенных с помощью диалогового окна данных.

**Компоненты OpenDialog и SaveDialog**

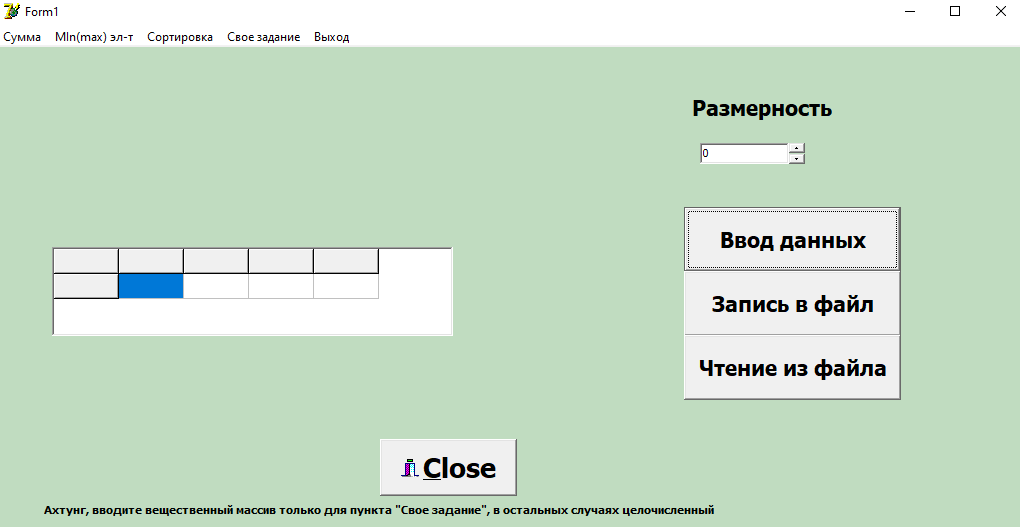
Эти компоненты имеют идентичны свойства:

FileName – содержит маршрут поиска и выбранный файл при успешном завершении диалога. Проверить существование файла можно функцией FileExists.

Filter – используется для отбора файлов, показываемых в диалоговом окне. При программном вводе фильтры задается одной длинной строкой, в которой символы | служат для разделения фильтруемых файлов от соответствующей маски выбора.

OpenDialog1.Filter:=’Текстовые файлы|\*.txt|Файлы Паскаля|\*.pas’ InitialDir – позволяет установить начальный каталог.

OpenDialog1.InitialDir:=’c:\program\borland\sourse’



**Рисунок 1 – пример формы приложения № 1**

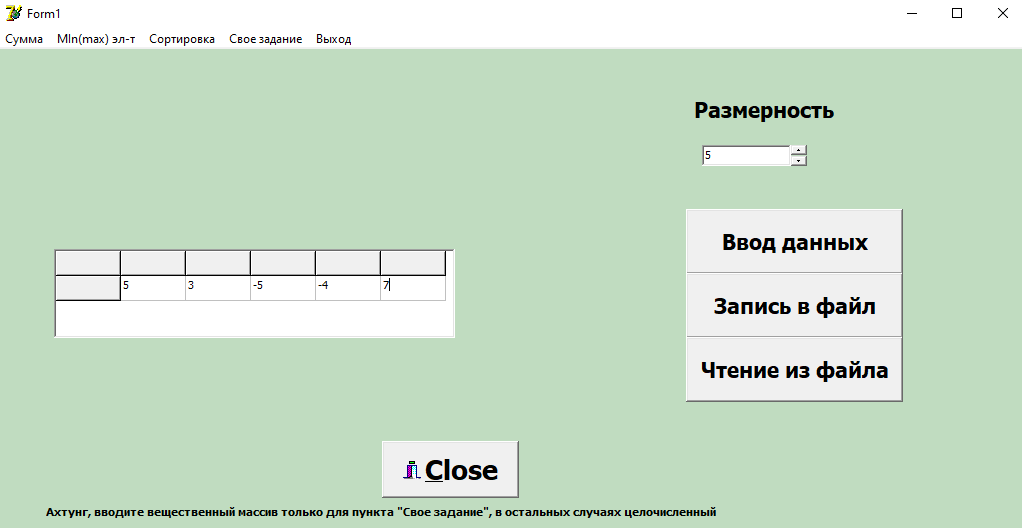
Вывод полученных результатов должен осуществляться на отдельную форму.

Должны быть обработаны все некорректные случаи ввода исходных данных. Все используемые в работе массивы должны быть ДИНАМИЧЕСКИМИ!!!

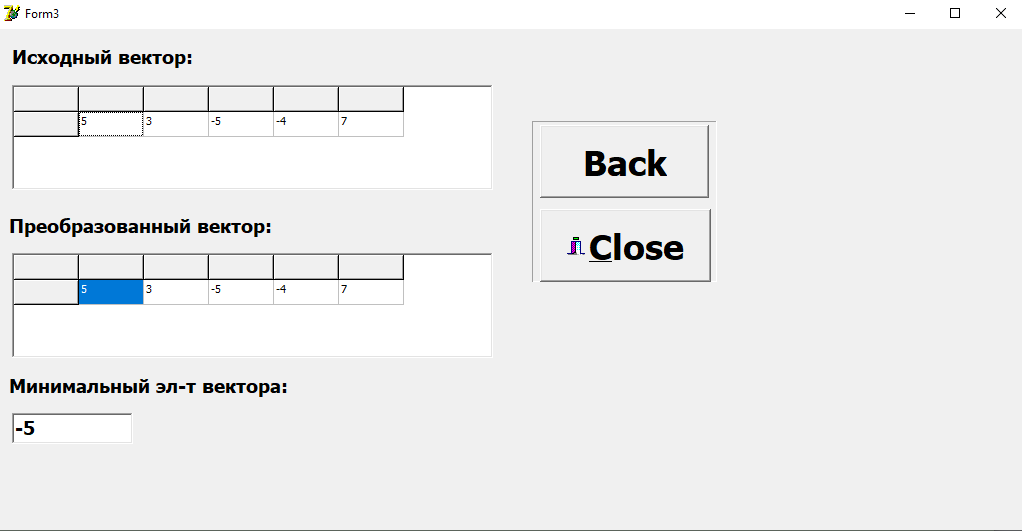
Вариант 23

1. Дан вещественный вектор *А(n)*. Получить новый вектор путем умножения элементов стоящих за максимальным элементом на минимальный элемент вектора. На печать выдавать исходный вектор, минимальный элемент и полученный вектор. н

На рисунке 2 изображена форма с первым тестовым примером для задания 1 варианта 14, на рисунке 3 изображена форма с полученными результатами для первого тестового примера для задания 1 варианта 14



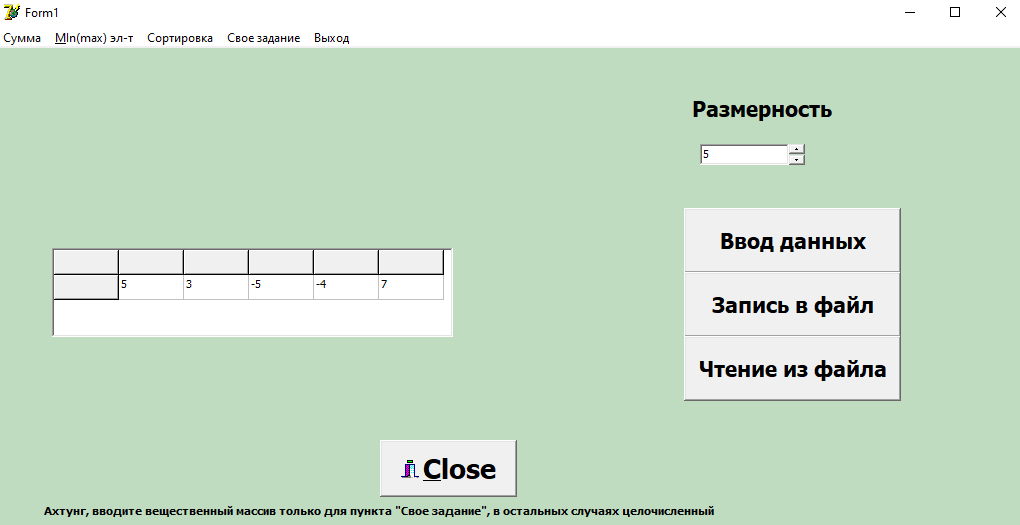
**Рисунок 2 – тестовый пример для задания**



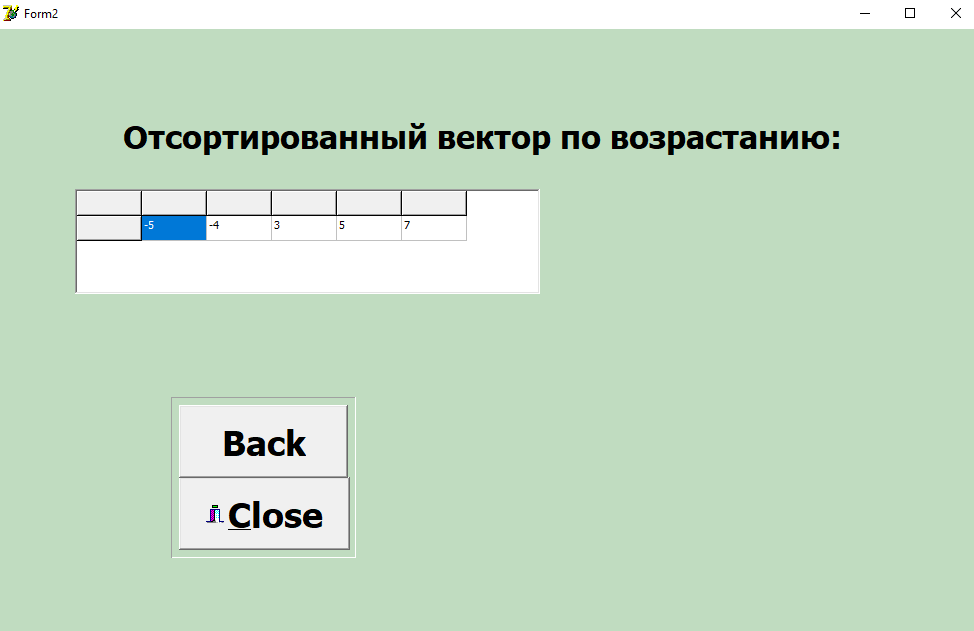
**Рисунок 3 – результат решения задания 1**

2) Дан целочисленный вектор *А(n).* Отсортировать методом линейного выбора с обменом.

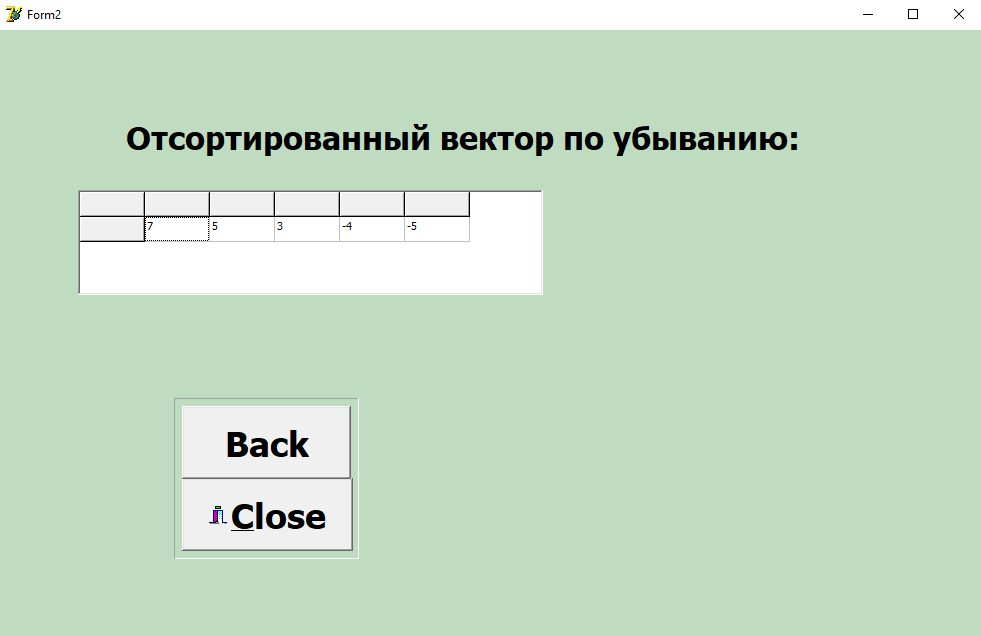
На рисунке 4 изображена форма с первым тестовым примером для задания 2 варианта 14, на рисунке 5 изображена форма с полученными результатами для первого тестового примера для задания 2 варианта 14, на рисунке 6 изображена форма с полученными результатами для второго тестового примера для задания 3 варианта 14,



**Рисунок 4 – тестовый пример для задания 2**



**Рисунок 5 – результат решения задания 2 (сортировка по возрастанию)**



**Рисунок 6 – результат решения задания 2 (сортировка по убыванию)**

# **2 Операции над двумерными массивами по заданным критериям**

Постановка задачи: разработать приложение для осуществления над матрицами следующих операций: сложение, вычитание, умножение матриц, умножение матрицы на число, сортировка (в соответствии со своим вариантом из лабораторной работы №6, нечётные варианты – сортировка элементов строк матрицы, чётные – столбцов; на отдельной вкладке – сортировка в соответствии с вариантом, см. ниже), транспонирование, нахождение обратной матрицы, подсчёт определителя матрицы. При разработке приложения использовать следующие компоненты: OpenDialog, SaveDialog, UpDown, PageControl, TrackBar. По нажатию соответствующих кнопок должны происходить запись и чтение из файла матриц и их размерностей, введённого числа. Должны быть обработаны все некорректные случаи ввода исходных данных.

**Страницы с закладками**

Для создания многостраничных окон используется компонент PageControl, который расположен на странице Win32. Первоначально компонент PageControl не содержит ни единой страницы. Для создания страницы необходимо щелкнуть правой кнопкой мыши на компоненте и выбрать в всплывающем меню команду New Page.

Свойства компоненты PageControl:

ActivePage – задает активную страницу.

Multiline – если равно True, то закладки располагаются в несколько рядов.

Pages – массив страниц.

PageCount – общее количество страниц.

Имена закладок изменяются в свойстве Caption.

**Компоненты страницы Dialog**

В состав Windows входит ряд стандартных типовых диалоговых окон, таких как окно выбора загружаемого файла, окно выбора шрифта и т.д. Работа со стандартными диалоговыми окнами осуществляется в три этапа.

Вначале на форму помещается соответствующий компонент и осуществляется настройка его свойств.

На втором этапе осуществляется вызов стандартного для диалогов метода Execute, который создает и показывает настроенное окно на экране. Вызов этого метода обычно располагается внутри обработчика какого-либо события. Например, обработчик выбора опции меню Открыть файл может вызвать метод Execute диалога ОpenDialog, обработчик нажатия кнопки Сохранить может вызвать такой же метод у компоненты SaveDialog. Только после обращения к Execute на экране появляется соответствующее диалоговое окно. Окно диалога является модальным окном, поэтому сразу после обращения к Execute дальнейшее выполнение программы приостанавливается до тех пор, пока пользователь не закроет окно. Функция Execute возвращает значение True, если результат диалога с пользователем был успешным.

Проанализировав результат Execute, программа может выполнить третий этап – использование введенных с помощью диалогового окна данных.

**Компоненты OpenDialog и SaveDialog** Эти компоненты имеют идентичные свойства:

FileName – содержит маршрут поиска и выбранный файл при успешном завершении диалога. Проверить существование файла можно функцией FileExists.

Filter – используется для отбора файлов, показываемых в диалоговом окне. При программном вводе фильтры задается одной длинной строкой, в которой символы | служат для разделения фильтруемых файлов от соответствующей маски выбора.

OpenDialog1.Filter:=’Текстовые файлы|\*.txt|Файлы Паскаля|\*.pas’ InitialDir – позволяет установить начальный каталог.

OpenDialog1.InitialDir:=’c:\program\borland\sourse’

Ниже приведены примеры процедур, осуществляющих запись и чтение из файла.

procedure TForm1.Button2Click(Sender: TObject); var f:TextFile; s:string; i,j,n:integer; begin

SaveDialog1.Filter:='Текстовые файлы|\*.txt|'; n:=strtoint(edit1.text); if SaveDialog1.Execute then begin

//открываем файл

AssignFile (f,SaveDialog1.Filename); Rewrite(f); writeln (f,n); for i:=1 to n do begin

for j:=1 to n do

writeln (f,stringgrid1.cells[i,j]); //работаем с векторами и матрицами !!! A[i] end;

CloseFile (f); end; end;

procedure TForm1.Button3Click(Sender: TObject); var i,j,n:integer; f:TextFile; s:string; begin

//n:=strtoint(edit1.text);

OpenDialog1.Filter:='Текстовые файлы|\*.txt|';

if ((OpenDialog1.Execute) and (FileExists(OpenDialog1.Filename))) then begin

//открываем файл

AssignFile

(f,OpenDialog1.Filename);

Reset(f); readln(f,n);

For i:=1 to n do for j:=1 to n do

begin readln (f,s);

stringgrid1.cells[i,j]:=s;

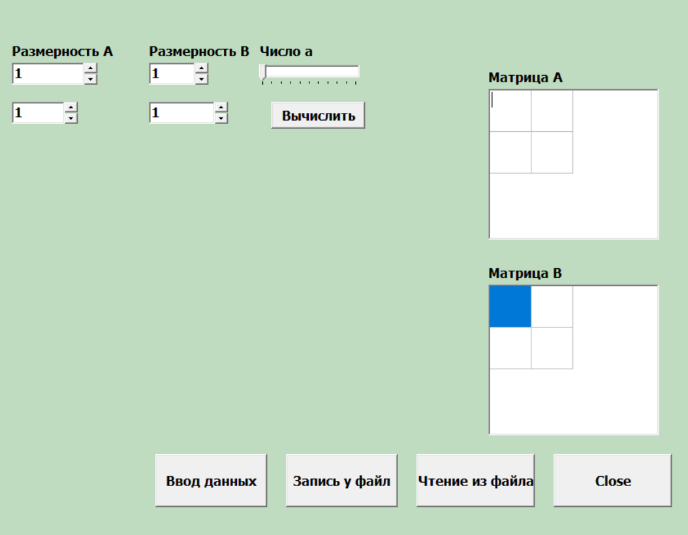
//работаем с векторами и матрицами !!! A[i] end;

CloseFile (f);

Stringgrid1.Visible:=True;

Edit1.Text:=inttostr(n);

Stringgrid1.ColCount:=n+1; end; end;

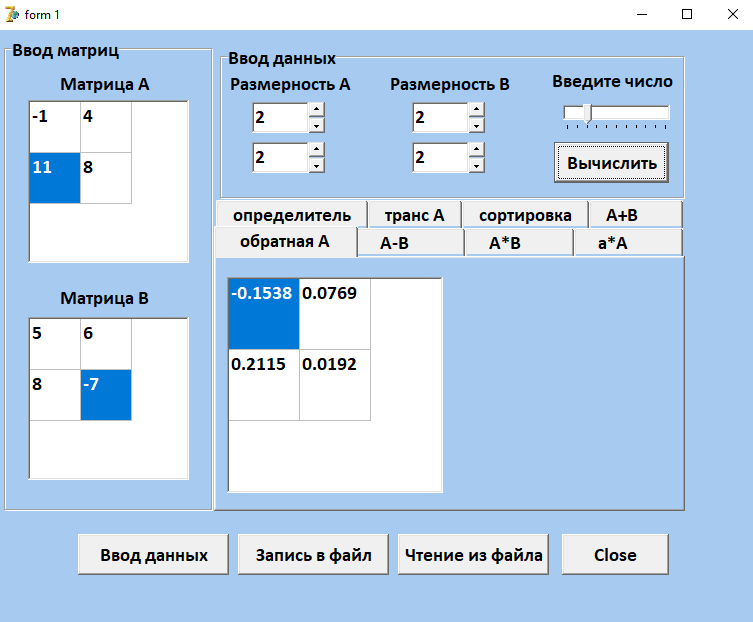


**Рисунок 7 – пример формы приложения № 2**

Вариант 23

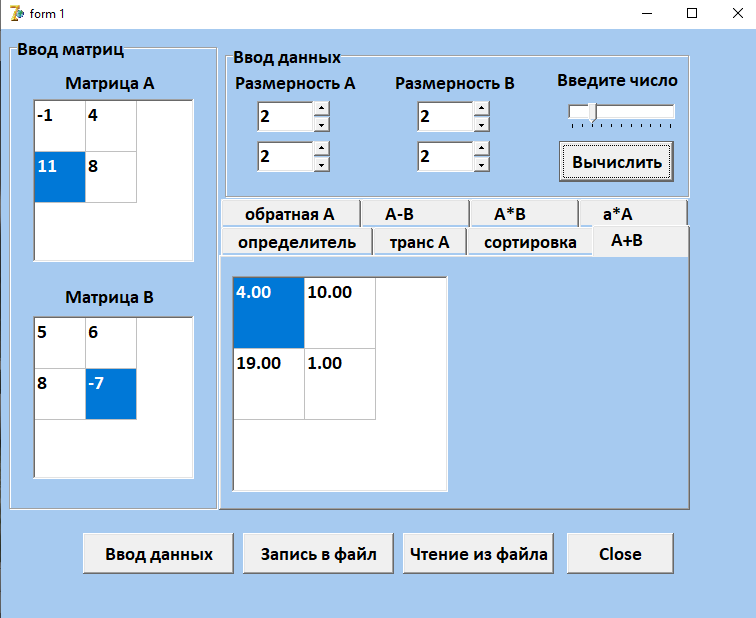
Осуществить сложение, вычитание, умножение матриц, умножение матрицы на число, сортировка (в соответствии с своим вариантом из лабораторной работы №14, нечётные варианты – сортировка элементов строк матрицы, чётные - столбцов), транспонирование, нахождение обратной матрицы, подсчёт определителя матрицы.

На рисунке 8 приведен пример формы приложения с вычисленной обратной матрицей для введенной матрицы размерности 2 на 2, на рисунке 9 приведен пример формы приложения сложения двух матриц размерности 2 на 2, на рисунке 10 приведен пример формы приложения вычитания двух матриц размерности 2 на 2, на рисунке 11 приведен пример формы приложения перемножения двух матриц размерности 2 на 2, на рисунке 12 приведен пример формы приложения умножения матрицы размерности 2 на 2, на действительное число, на рисунке 13 приведен пример транспонирования введенной матрицы 2 на 2, на рисунке 14 приведен пример сортировки матрицы размерности 2 на 2, на рисунке 15 приведен пример вычисления определителя матрицы 2 на 2.

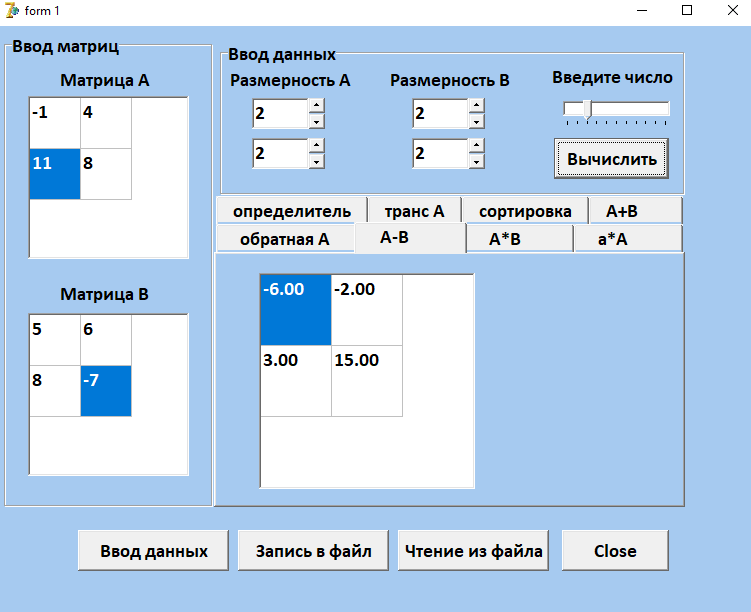


**Рисунок 8 – пример нахождения обратной матрицы**

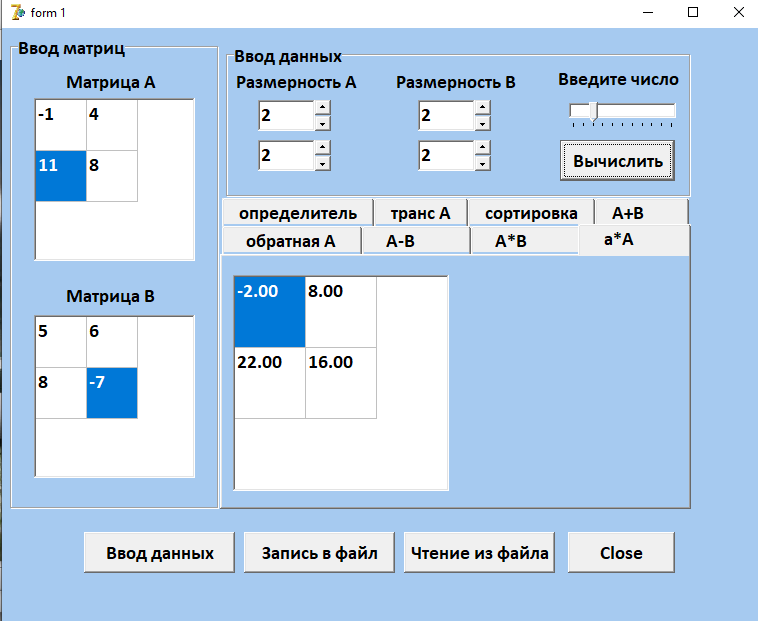
**Рисунок 9 – пример сложения двух матриц**

****

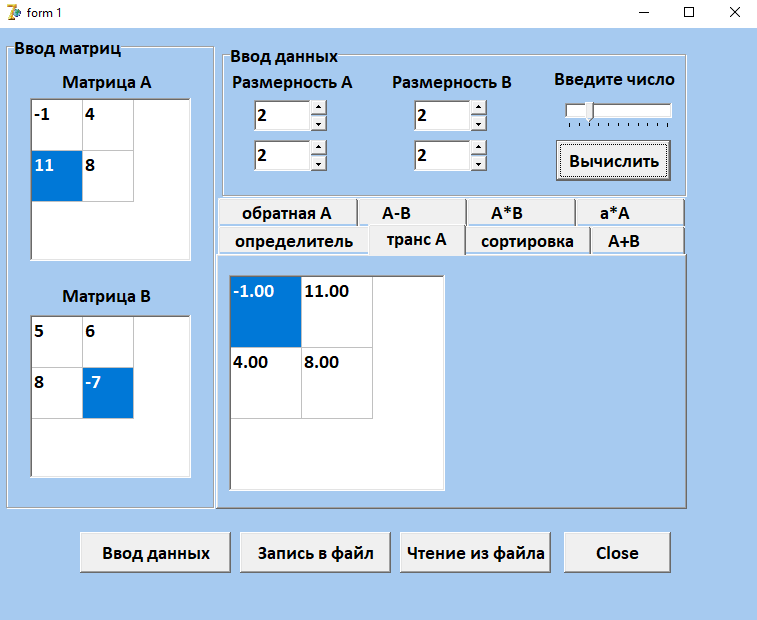
**Рисунок 10 – пример вычитания двух матриц**

****

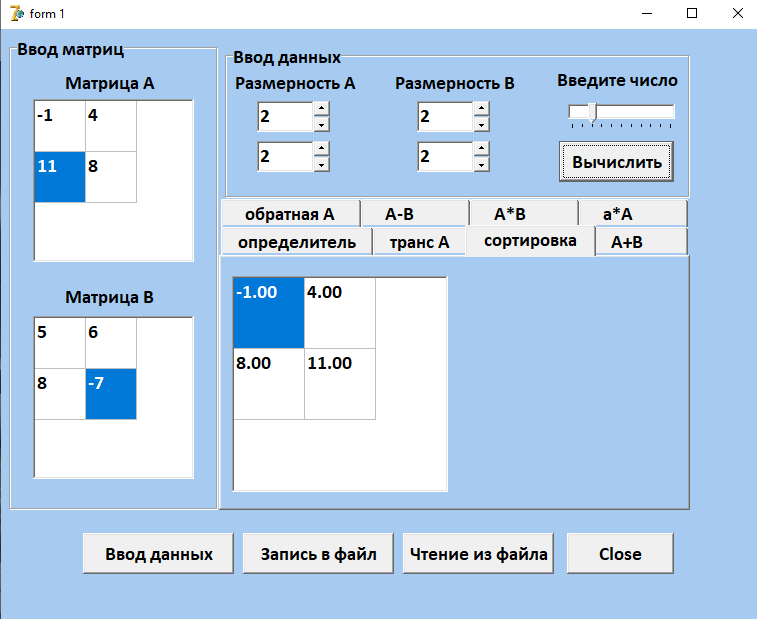
**Рисунок 11 – пример перемножения двух матриц**

****

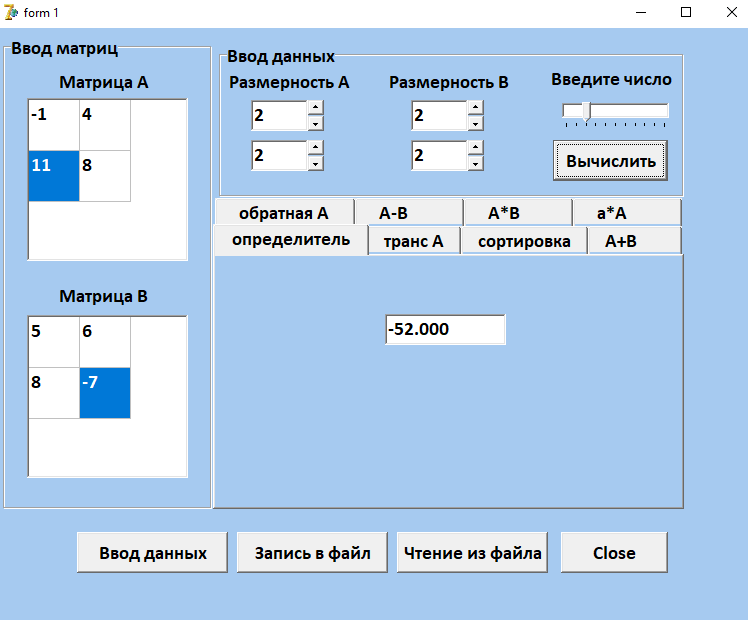
**Рисунок 12 – пример умножения матрицы на действительное число**

****

**Рисунок 13 – пример транспонирования матрицы**

****

**Рисунок 14 – пример сортировки по варианту (матрицы по строкам)**

****

**Рисунок 15 – пример вычисления определителя матрицы**

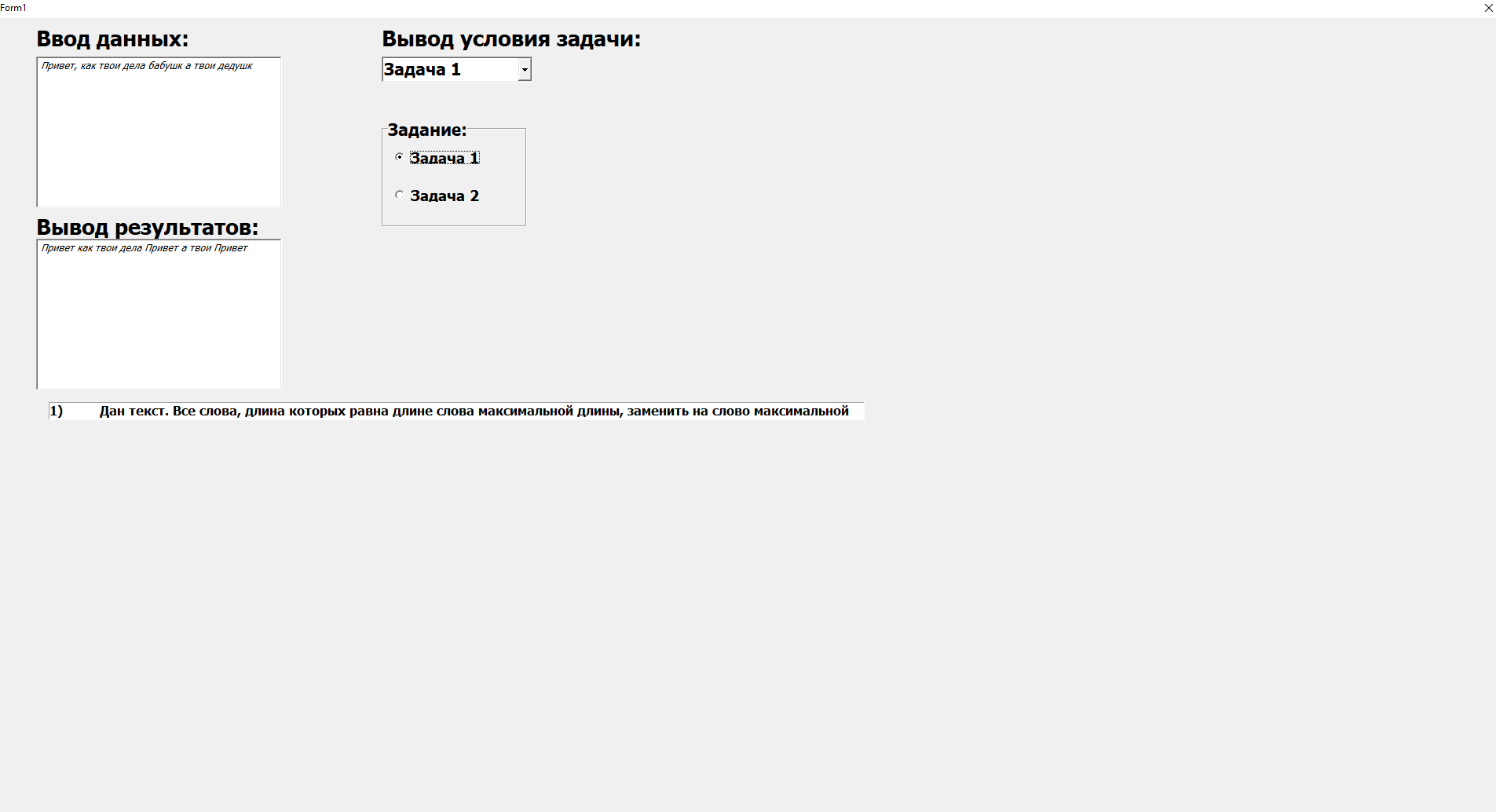
# **3 Обработка символьной информации**

Постановка задачи: разработать приложение обработки строк. Использовать компоненты Memo, ListBox, ComboBox, организовать ввод строки в компоненту Memo, а вывод результатов в ListBox. Условия задач выводить в компоненте StaticText. Организовать переключение задач с помощью RadioButton и RadioGroup (группы радиокнопок). Выбор задач осуществить с помощью ComboBox. Использовать кнопки BitBtn для выхода, записи и чтения исходной строковой информации.

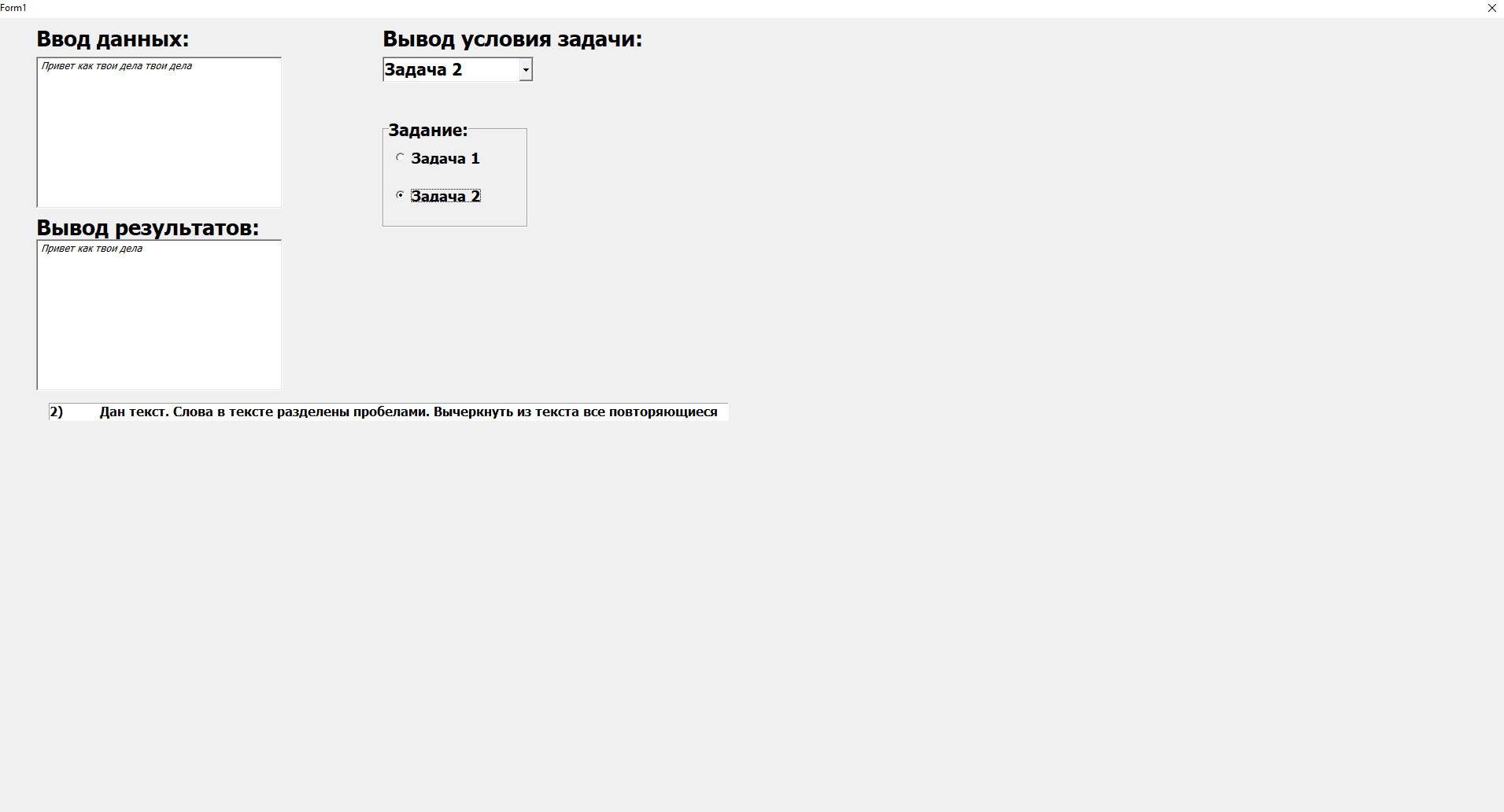
Вариант 23

1. Дан текст. Все слова, длина которых равна длине слова максимальной длины, заменить на слово максимальной длины.
2. Дан текст. Слова в тексте разделены пробелами. Вычеркнуть из текста все повторяющиеся слова.

На рисунке 16 приведен пример формы приложения №3 с тестовым примером для задания 1, на рисунке 17 приведен пример формы приложения №3 с тестовым примером для задания 2.



**Рисунок 16 – тестовый пример для задания 1**

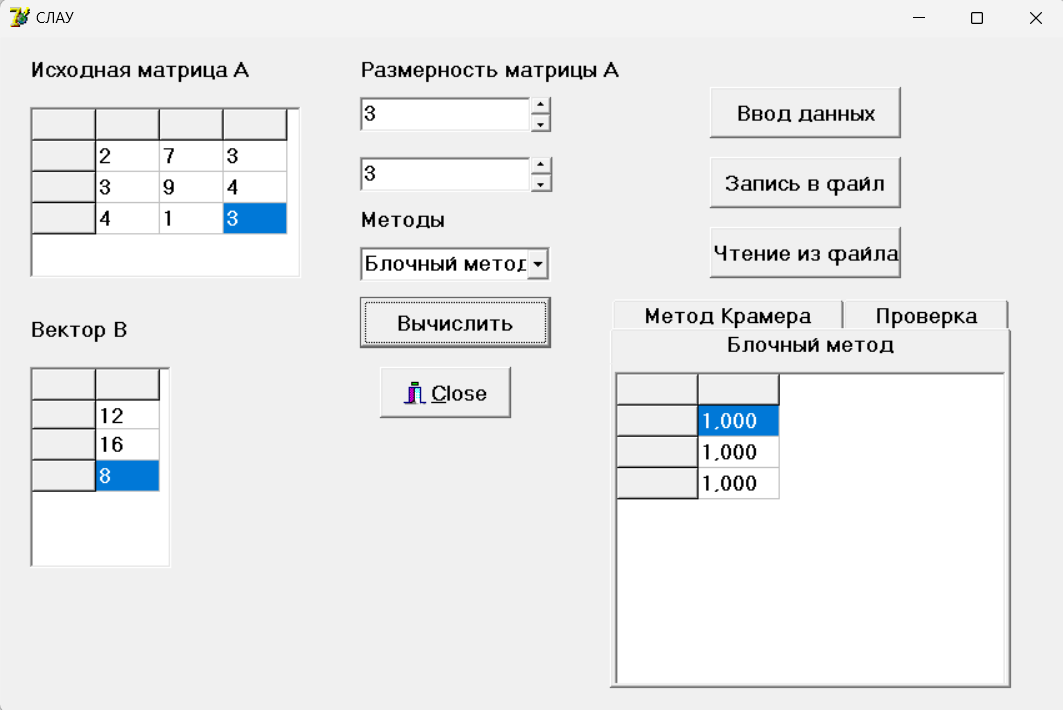
****

**Рисунок 17 – тестовый пример для задания**

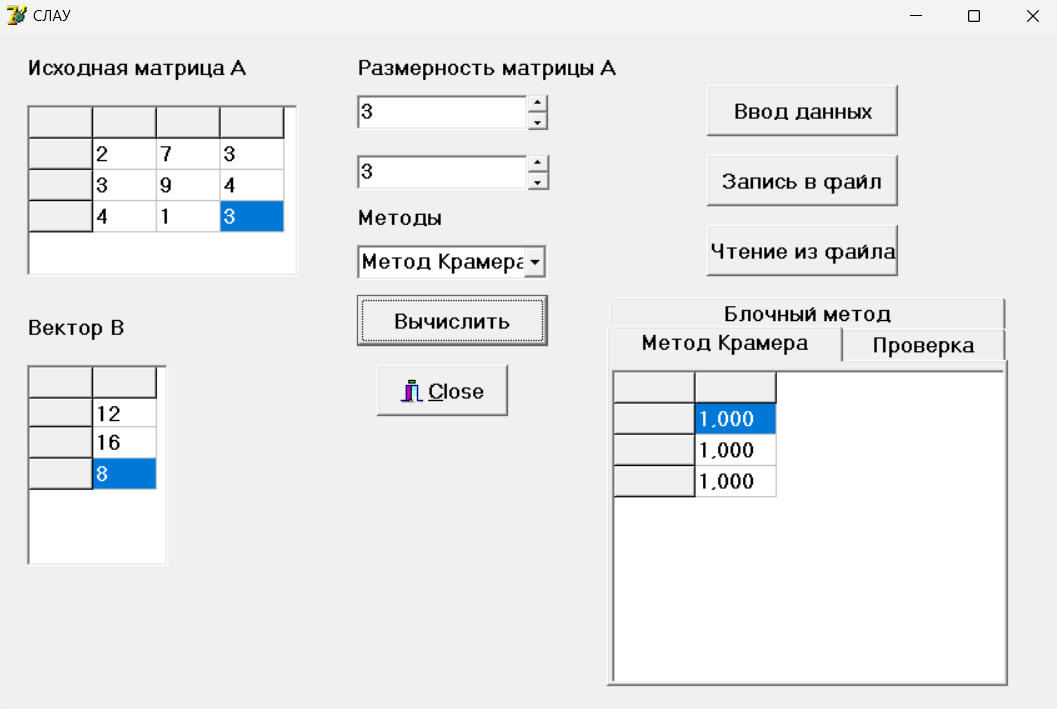
# **4 Методы решения систем линейных алгебраических уравнений**

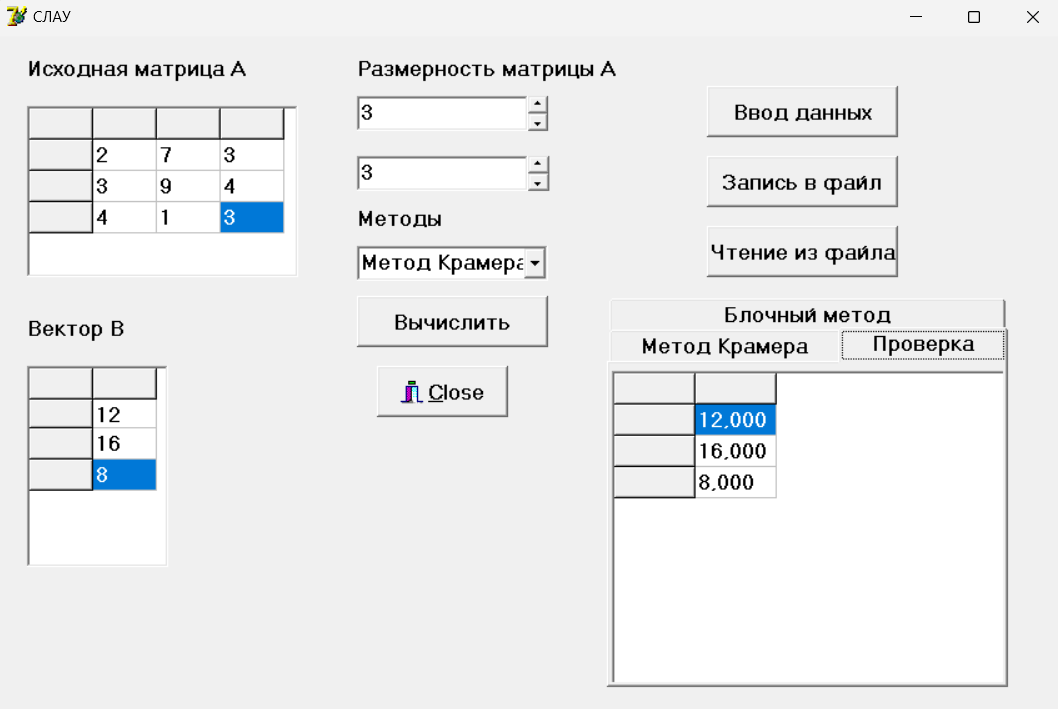
Постановка задачи. Запрограммировать матричный (блочный) метод и один из методов решения систем алгебраических уравнений согласно своему варианту. Программа должна обрабатывать все возможные ошибки, должно быть организовано чтение и запись в файл необходимых данных. Должна быть организована проверка полученных результатов. Необходимые вычисления (свой метод решения СЛАУ, перемножение матриц, векторов и. т. д.) должны быть организованы в отдельных процедурах или функциях. Нечетные варианты – метод Гаусса, четные – метод Крамера.

На рисунке 18 приведен пример формы приложения №4 с тестовым примером для решения СЛАУ блочным методом, на рисунке 19 приведен пример формы приложения №4 с тестовым примером для решения СЛАУ методом Крамера, на рисунке 20 приведен пример формы приложения №4 с тестовым примером выполнения проверки решения СЛАУ.



**Рисунок 18 – пример решения СЛАУ блочным методом**

**Рисунок 19 – пример решения СЛАУ методом Крамера**



**Рисунок 20 – пример выполнения проверки решения СЛАУ**

# **5 Теоретическая часть**

**5.1** Порядок работы с типизированными файлами в Delphi:

1. Действия с элементами массива:

Вы можете использовать циклы и математические операции для выполнения действий с элементами массива, таких как поиск сумм и произведений. Например, чтобы найти сумму элементов массива, вы можете использовать цикл для перебора всех элементов и суммирования их значений. Аналогично, вы можете использовать циклы и математические операции умножения, чтобы найти произведение элементов массива.

var

arr: array[1..10] of integer;

sum, i: integer;

begin sum := 0;

for i := 1 to 10 do

begin read(arr[i], 'Enter element #');

sum := sum + arr[i];

end;

writeln('Sum of array elements: ', sum);

end.

(Пример поиска суммы элементов массива на паскале)

1. Главная и боковые диагонали:

Чтобы найти главную и боковые диагонали массива, вы можете использовать индекс элементов массива. Главная диагональ перемещается из верхнего левого угла в нижний правый угол, а боковая диагональ перемещается из верхнего правого угла в нижний левый угол. Чтобы найти эти значения, вы можете использовать соответствующий индекс массива.

var

arr: array[1..10] of integer;

sum, i: integer;

begin sum := 0;

for i := 1 to 10 do

begin read(arr[i], 'Enter element #');

sum := sum + arr[i];

end;

writeln('Sum of array elements: ', sum);

end.

Var

i, j, sum: integer;

matrix: array[1..10, 1..10] of integer;

begin

for i := 1 to 10 do

for j := 1 to 10 do

matrix[i, j] := random(100);

sum := 0;

for i := 1 to 10 do sum := sum + matrix[i, i]; writeln('Сумма элементов главной диагонали:', sum);

end.

(Пример поиска суммы элементов на главной диагонали)

**5.2** Пять уровней инкапсуляции. Вызов родительских методов

Инкапсуляция - фундаментальный концепт объектно-ориентированного программирования (ООП), который позволяет разработчикам скрыть детали реализации объекта от внешнего мира, оставляя доступ к функциональности объекта только через управляемый интерфейс. В Delphi инкапсуляция достигается с помощью классов, методов и свойств.

1. Пять уровней инкапсуляции в Delphi:
   1. Public:

Члены класса находящиеся в данном разделе доступны из любой точки программы.

1.2. Раздел private:

Этот раздел накладывает самые жесткие ограничения на видимость указанных в нем членов класса. Они доступны только в том модуле, где данный класс описан. Как правило, все поля класса помещаются в эту секцию.

1.3. Раздел Protected:

Видимость членов класса расположенных в данном разделе совпадает с видимостью раздела Private, но с единственным отличием. Члены класса раздела Protected доступны также внутри методов класса, являющихся наследниками данного класса и описаных в других модулях.

1.4. Раздел Published:

В этом разделе располагаются свойства класса: т.е. поля доступные для редактирования и изменения значений во время проектирования приложения из инспектора объектов. По видимости свойства не отличаются от членов классов описанных в разделе Public. Если перед полями нет ключевого слова, то подразумевается слово Published.

1.5 Раздел automated:

Правила видимости членов раздела совпадает с правилами видимости для раздела Public. Описание разрешается размещать в данном разделе только, если он является наследником стандартного класса TAutoObject, предназначенного для создания так называемых серверов автоматизации по технологии COM.

1. Вызов родительских методов:

Чтобы вызвать одноименный метод ближайшего родительского класса необходимо в нужном месте программы указать ключевое слово inherited.В момент, когда программа встретит это слова работа метода временно прервется и вызовется одноименный метод родительского класса. Если требуется вызвать другой метод, не совпадающий по названию с текущим, то его можно указать после ключевого слова явно inherited Click;

# ЗАКЛЮЧЕНИЕ

В курсовой работе были рассмотрены основные алгоритмы обработки одномерных массивов, основные алгоритмы обработки двумерных массивов, основные алгоритмы обработки символьной информации, основные алгоритмы решения систем линейных алгебраических уравнений. Для этого были изучены следующие вопросы: основные операции для работы с одномерными массивами, такие как инициализация, сортировка, поиск и манипуляции с элементами массива; основы работы с матрицами, включая операции сложения, умножения, транспонирования, определения детерминанта, нахождения обратной матрицы; рассмотрение методов кодирования, поиска и замены подстрок в строках; изучение методов решения систем линейных алгебраических уравнений, таких как метод Гаусса, Крамера, блочный метод. Были разработаны приложения, реализующие решение поставленных задач на обработку одномерных массивов, двумерных массивов, строковой информации и решений систем линейных алгебраических уравнений. Информация о реализованных заданиях, согласно варианту, находится в соответствующем разделе, программный код приложения обработки одномерных массивов находится в приложении А, программный код приложения обработки двумерных массивов – в приложении Б, программный код приложения обработки строк – в приложении В, программный код приложения решения СЛАУ – в приложении Г, скриншот проверки на плагиат – в приложении Д.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бобровский, С. И. Delphi7: учебный курс / С. И. Бобровский. – СПб.: Питер, 2008. – 736 с.

2. Культин, Н. Б. Основы программирования в Delphi7 / Н. Б. Культин. – СПб.: БХВ-Петербург, 2003. – 638 с.

3. Стивенс, Р. Delphi. Готовые алгоритмы / Р. Стивенс; пер. с англ. Мерещука П.А. – 2-е изд. стер. – М.: ДМК Пресс; СПб.: Питер, 2004. – 384с.

4. Сухарев, М. В. Основы Delphi. Профессиональный подход / М. В. Сухарев. – СПб.: Наука и техника, 2004. – 614 с.

5. Фаронов, В. В. Delphi 5. Учебный курс / В.В. Фаронов – М.: «Нолидж», 2000. – 608 с.

6. Фаронов, В.В. Программирование баз данных в Delphi 7. Учебный курс / В.В. Фаронов. – СПб.: Питер, 2006. – 459 с.

7. Основы алгоритмизации и программирования: учебно-методический комплекс / сост.: Д. С. Кузьменков, М. С. Долинский. – Гомель: ГГУ имени Ф. Скорины, 2022. – 168 с. – Режим доступа: http://dot3.gsu.by/course/view.php?id=5186

8. Кононова, З. А. Программирование в Delphi: создание приложений : учебное пособие / З. А. Кононова, С. О. Алтухова. – Липецк: Липецкий государственный педагогический университет им. П. П. Семенова-Тян-Шанского, 2020. – Часть 2. – 87 с. – Режим доступа : по подписке : <https://biblioclub.ru/index.php?page=book&id=619369>

9. Гайрабекова, Т. И. Алгоритмы и алгоритмические языки : учебное пособие / Т. И. Гайрабекова. – Грозный: ЧГУ, 2021. – 133 с. – Режим доступа: для авториз. пользователей: https://e.lanbook.com/book/264020.

# ПРИЛОЖЕНИЕ А

# 

# Код программы для обработки одномерных массивов

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ComCtrls, Grids, Buttons, Menus;

type mas = array of integer;

masf = array of Real;

type

TForm1 = class(TForm)

btn1: TBitBtn;

SG: TStringGrid;

baton1: TBitBtn;

baton2: TBitBtn;

baton3: TBitBtn;

lbl1: TLabel;

ud1: TUpDown;

e1: TEdit;

main: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

iN3: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N101: TMenuItem;

N102: TMenuItem;

N6: TMenuItem;

N7: TMenuItem;

M: TMenuItem;

Sort: TMenuItem;

my: TMenuItem;

Exit: TMenuItem;

max1: TMenuItem;

min1: TMenuItem;

Strue: TMenuItem;

Sreve: TMenuItem;

mine: TMenuItem;

OpenDialog1: TOpenDialog;

SaveDialog1: TSaveDialog;

l2: TLabel;

procedure baton1Click(Sender: TObject);

procedure FormClick(Sender: TObject);

procedure Float(var Key: Char);

procedure N2Click(Sender: TObject);

procedure iN3Click(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure N4Click(Sender: TObject);

procedure N5Click(Sender: TObject);

procedure N101Click(Sender: TObject);

procedure N102Click(Sender: TObject);

procedure N6Click(Sender: TObject);

procedure N7Click(Sender: TObject);

procedure max1Click(Sender: TObject);

procedure min1Click(Sender: TObject);

procedure StrueClick(Sender: TObject);

procedure SreveClick(Sender: TObject);

procedure FormActivate(Sender: TObject);

procedure mineClick(Sender: TObject);

procedure baton2Click(Sender: TObject);

procedure baton3Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

A:mas;

b: masf;

n: Integer;

implementation

uses Unit2, Unit3;

function E(var A: mas;var n: Integer):boolean;

var i:longint;

begin

n:=strtoint(Form1.E1.text);

SetLength(A,n+1);

for i:=1 to n do

begin

if Form1.SG.Cells[i,1]='' then

begin

MessageDlg('Заполнен не весь SG!',mtError,[MbOk],0);

MessageDlg('Заполните!',mtError,[MbOk],0);

E:=false;

Exit;

end

else

begin

A[i]:=StrToInt(Form1.SG.Cells[i,1]);

E:=true;

end;

end;

end;

function v(var A: masf;var n: Integer):boolean;

var i:longint;

begin

n:=strtoint(Form1.E1.text);

SetLength(A,n+1);

for i:=1 to n do

begin

if Form1.SG.Cells[i,1]='' then

begin

MessageDlg('Заполнен не весь SG!',mtError,[MbOk],0);

MessageDlg('Заполните!',mtError,[MbOk],0);

v:=false;

Exit;

end

else

begin

A[i]:=StrToFloat(Form1.SG.Cells[i,1]);

v:=true;

end;

end;

end;

procedure TForm1.Float(var Key: Char);

var key1:string;i,n:integer;

begin

n:=strtoint(e1.text);

key1:='1234567890';

for i:=1 to n do

begin

If not (Key in ['0'..'9',#8,#45]) then

Key:=#0;

end;

end;

{$R \*.dfm}

procedure TForm1.baton1Click(Sender: TObject);

begin

lbl1.Visible:=boolean(1);

e1.Visible:=Boolean(1);

ud1.Visible:=True;

end;

procedure TForm1.FormClick(Sender: TObject);

var n,i: Integer;

begin

SG.Visible:=Boolean(1);

SG.Options:=SG.Options+[goEditing];

n:=strtoint(e1.Text);

for i:=1 to n do

begin

SG.ColCount:=i+1;

SG.RowCount:=2;

end;

l2.Visible:=True;

end;

procedure TForm1.N2Click(Sender: TObject);

var i,j,sum,n: Integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.lb.Caption:='Сумма всех элементов';

sum:=0;

for i:=1 to n do

inc(sum,a[i]);

Form2.e2.Text:=inttostr(sum);

end;

end;

procedure TForm1.iN3Click(Sender: TObject);

var n,i,sum,j: integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.lb.Caption:='Сумма всех отрицательных эл-тов:';

for j:=1 to SG.ColCount-1 do

A[j]:=StrToInt(SG.Cells[j,1]);

for i:=1 to n do

if a[i]<0 then

Inc(sum,a[i]);

form2.e2.Text:=IntToStr(sum);

end;

end;

procedure TForm1.N3Click(Sender: TObject);

var n,i,j,sum: integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.lb.Caption:='Сумма всех четных эл-тов:';

sum:=0;

for i:=1 to n do

if a[i] mod 2 = 0 then

inc(sum,A[i]);

form2.e2.Text:=inttostr(sum);

end;

end;

procedure TForm1.N4Click(Sender: TObject);

var n,i,j,sum: integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.lb.Caption:='Сумма всех нечетных эл-тов:';

sum:=0;

for i:=1 to n do

if a[i] mod 2 <> 0 then

inc(sum,A[i]);

form2.e2.Text:=inttostr(sum);

end;

end;

procedure TForm1.N5Click(Sender: TObject);

var i,j,n,sum: Integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.lb.Caption:='Сумма всех положительных эл-тов:';

sum:=0;

for i:=1 to n do

if a[i]>0 then

Inc(sum,a[i]);

form2.e2.Text:=IntToStr(sum);

end;

end;

procedure TForm1.N101Click(Sender: TObject);

var i,j,sum,n: integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.lb.Caption:='Сумма эл-тов значение которых >10';

sum:=0;

For i:=1 to n do

if A[i]>10 then

inc(sum,A[i]);

Form2.e2.Text:=IntToStr(sum);

end;

end;

procedure TForm1.N102Click(Sender: TObject);

var i,j,sum,n: integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.lb.Caption:='Сумма эл-тов значение которых <10';

sum:=0;

For i:=1 to n do

if A[i]<10 then

inc(sum,A[i]);

Form2.e2.Text:=IntToStr(sum);

end;

end;

procedure TForm1.N6Click(Sender: TObject);

var i,j,n,sum,srsum: integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.Width:=1000;

Form2.lb.Caption:='Сумма эл-тов значение которых больше сред. арифм.';

sum:=0;

for i:=1 to n do

inc(sum,A[i]);

srsum:= sum div n ;

sum:=0;

for i:=1 to n do

if A[i]>srsum then

inc(sum,A[i]);

Form2.e2.Text:=inttostr(sum);

end;

end;

procedure TForm1.N7Click(Sender: TObject);

var

i,j,n,srsum,sum: Integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.Width:=1000;

Form2.lb.Caption:='Сумма эл-тов значение которых меньше сред. арифм.';

sum:=0;

for i:=1 to n do

inc(sum,A[i]);

srsum:= sum div n ;

sum:=0;

for i:=1 to n do

if A[i]<srsum then

inc(sum,A[i]);

Form2.e2.Text:=inttostr(sum);

end;

end;

procedure TForm1.max1Click(Sender: TObject);

var i,j,n,max,ii: Integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Width:=1000;

Form2.Show;

Form2.e22.Visible:=true;

Form2.lb.Caption:='Значение максимального эл-та вектора:';

Form2.lb1.Visible:=true;

Form2.lb1.Caption:='Индекс максимального эл-та в векторе:';

max:=a[1];

ii:=1;

for i:=1 to n do

if a[i]>max then

begin

max:=A[i];

ii:=i;

end;

Form2.e2.Text:=IntToStr(max);

Form2.e22.Text:=IntToStr(ii);

end;

end;

procedure TForm1.min1Click(Sender: TObject);

var i,j,n,max,ii: Integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Width:=1000;

Form2.Show;

Form2.e22.Visible:=true;

Form2.lb.Caption:='Значение минимального эл-та вектора:';

Form2.lb1.Visible:=true;

Form2.lb1.Caption:='Индекс минимально эл-та в векторе:';

max:=a[1];

ii:=1;

for i:=1 to n do

if a[i]<max then

begin

max:=A[i];

ii:=i;

end;

Form2.e2.Text:=IntToStr(max);

Form2.e22.Text:=IntToStr(ii);

end;

end;

procedure TForm1.StrueClick(Sender: TObject);

var i,j,n,temp,key: Integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.Width:=1000;

Form2.SG1.Visible:=True;

Form2.lb.Caption:='Отсортированный вектор по возрастанию:';

for i:=1 to n do

begin

Form2.SG1.ColCount:=i+1;

Form2.SG1.RowCount:=2;

end;

for i := 2 to n do

begin

key := A[i];

j := i;

while (j > 1) and (A[j - 1] > key) do

begin

{обмен элементов}

temp := A[j];

A[j] := A[j - 1];

A[j - 1] := temp;

j := j - 1;

end;

A[j] := key;

end;

for j:=1 to Form2.SG1.ColCount do

Form2.SG1.Cells[j,1]:=IntToStr(A[j]);

end;

end;

procedure TForm1.SreveClick(Sender: TObject);

var i,j,n,temp,key: Integer;

begin

n:=StrToInt(e1.Text);

if E(A,n)=true then begin

Form2.Show;

Form2.Width:=1000;

Form2.lb.Caption:='Отсортированный вектор по убыванию:';

Form2.SG1.Visible:=True;

for i:=1 to n do

begin

Form2.SG1.ColCount:=i+1;

Form2.SG1.RowCount:=2;

end;

for i := 2 to n do

begin

key := A[i];

j := i;

while (j > 1) and (A[j - 1] < key) do

begin

temp := A[j];

A[j] := A[j - 1];

A[j - 1] := temp;

j := j - 1;

end;

A[j] := key;

end;

For j:=1 to Form2.SG1.ColCount do

Form2.SG1.Cells[j,1]:=IntToStr(A[j]);

end;

end;

procedure TForm1.FormActivate(Sender: TObject);

begin

Form1.Color:=clMoneyGreen;

end;

procedure TForm1.mineClick(Sender: TObject);

var i,j,n,m,temp,im,imax: Integer;

max,min: real;

begin

n:=StrToInt(e1.Text);

m:=StrToInt(e1.Text);

if v(b,n)=true then begin

Form3.Show;

for i:=1 to n do

begin

Form3.SG1.ColCount:=i+1;

Form3.SG1.RowCount:=2;

end;

for i:=1 to n do

begin

Form3.SG2.ColCount:=i+1;

Form3.SG2.RowCount:=2;

end;

for i:=1 to n do

Form3.SG1.Cells[i,1]:=FloatToStr(b[i]);

max:=b[1];

min:=b[1];

for i:=1 to n do

begin

if (b[i]>max) then

max:=b[i]

else if b[i]<min then

min:=b[i];

end;

for i:=1 to n do

if b[i]=max then

begin

imax:=i;

Break;

end;

for i:=imax+1 to n do

begin

b[i]:=b[i]\*min;

end;

Form3.e1.Text:=FloatToStr(min);

for i:=1 to n do

Form3.SG2.Cells[i,1]:=FloatToStr(b[i]);

end;

end;

procedure TForm1.baton2Click(Sender: TObject);

var

f:TextFile;

s:string;

i,j,n:integer;

begin

SaveDialog1.Filter:='Текстовые файлы|\*.txt|';

n:=strtoint(e1.text);

if SaveDialog1.Execute then

begin

AssignFile (f,SaveDialog1.Filename);

Rewrite(f);

writeln (f,n);

for i:=1 to n do

writeln (f,SG.cells[j,1]);

CloseFile (f);

end;

end;

procedure TForm1.baton3Click(Sender: TObject);

var

i,j,n:integer;

f:TextFile;

s:string;

begin

OpenDialog1.Filter:='Текстовые файлы|\*.txt|';

if OpenDialog1.Execute and FileExists(OpenDialog1.Filename) then

begin

AssignFile (f,OpenDialog1.Filename);

Reset(f);

readln(f,n);

for i:=1 to n do

begin

readln (f,s);

SG.cells[1,j]:=s;

end;

CloseFile (f);

SG.Visible:=True;

e1.Text:=inttostr(n);

end;

end;

end.1;

# 

# ПРИЛОЖЕНИЕ Б

**Код программы для обработки двумерных массивов**

**unit** Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Grids, ComCtrls, StdCtrls, Buttons, jpeg, ExtCtrls;

type

arr\_type = array of array of Real;

TForm1 = class(TForm)

matrix\_a\_sg: TStringGrid;

matrix\_b\_sg: TStringGrid;

nums\_trackbar: TTrackBar;

write\_to\_file\_btn: TBitBtn;

read\_from\_file\_btn: TBitBtn;

enter\_values\_btn: TBitBtn;

calc\_btn: TBitBtn;

len\_of\_arr\_a: TEdit;

width\_of\_arr\_a: TEdit;

len\_of\_arr\_b: TEdit;

width\_of\_arr\_b: TEdit;

up\_down\_len\_a: TUpDown;

up\_down\_width\_a: TUpDown;

up\_down\_len\_b: TUpDown;

up\_down\_width\_b: TUpDown;

close\_btn: TBitBtn;

dlgOpen: TOpenDialog;

dlgSave: TSaveDialog;

matrix\_a\_lbl: TLabel;

matrix\_b\_lbl: TLabel;

col\_count\_a\_lbl: TLabel;

col\_count\_b\_lbl: TLabel;

n\_count\_lbl: TLabel;

page: TPageControl;

Multiply: TTabSheet;

error\_inversed: TLabel;

inversed\_a\_sg: TStringGrid;

a\_minus\_b: TTabSheet;

error\_a\_minus\_b: TLabel;

a\_minus\_b\_sg: TStringGrid;

a\_mult\_b: TTabSheet;

error\_a\_mult\_b: TLabel;

a\_mult\_b\_sg: TStringGrid;

num\_mult\_a: TTabSheet;

a\_mult\_by\_num: TStringGrid;

det\_a: TTabSheet;

det\_a\_edt: TEdit;

transpose\_a\_page: TTabSheet;

transposed\_a\_sg: TStringGrid;

sorted\_a: TTabSheet;

sorted\_a\_sg: TStringGrid;

a\_plus\_b: TTabSheet;

error\_a\_plus\_b: TLabel;

a\_plus\_b\_sg: TStringGrid;

det\_error\_lbl: TLabel;

grp1: TGroupBox;

grp2: TGroupBox;

procedure up\_down\_len\_aClick(Sender: TObject; Button: TUDBtnType);

procedure up\_down\_width\_aClick(Sender: TObject; Button: TUDBtnType);

procedure up\_down\_len\_bClick(Sender: TObject; Button: TUDBtnType);

procedure up\_down\_width\_bClick(Sender: TObject; Button: TUDBtnType);

procedure close\_btnClick(Sender: TObject);

procedure calc\_btnClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure read\_from\_file\_btnClick(Sender: TObject);

procedure write\_to\_file\_btnClick(Sender: TObject);

procedure enter\_values\_btnClick(Sender: TObject);

procedure FormClick(Sender: TObject);

procedure matrix\_a\_sgKeyPress(Sender: TObject; var Key: Char);

procedure matrix\_b\_sgKeyPress(Sender: TObject; var Key: Char);

procedure bgClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

type mas = array of array of Real;

var

Form1: TForm1;

a, b: arr\_type;

len\_a, len\_b, width\_a, width\_b: Integer;

result: Real;

implementation

uses Unit2;

{$R \*.dfm}

function init(): Boolean;

var i, j: integer; ADlg: TForm; Ctrl: TControl;

begin

for i:=0 to len\_a-1 do // Проверка заполненности матрицы А

for j:=0 to width\_a-1 do

begin

if Form1.matrix\_a\_sg.Cells[i, j] = '' then

begin

ADlg := CreateMessageDialog('Заполните матрицу А', mtError, [mbOK]);

with Adlg do

try

Caption := 'Ошибка!';

Ctrl := FindChildControl('Yes');

Case ShowModal of

ID\_YES: ;

end;

finally

Free;

end;

init := False;

Exit;

end;

end;

for i:=0 to len\_b-1 do // Проверка заполненности матрицы В

for j:=0 to width\_b-1 do

begin

if Form1.matrix\_b\_sg.Cells[i, j] = '' then

begin

ADlg := CreateMessageDialog('Заполните матрицу В', mtError, [mbOK]);

with Adlg do

try

Caption := 'Ошибка!';

Ctrl := FindChildControl('Yes');

Case ShowModal of

ID\_YES: ;

end;

finally

Free;

end;

init := False;

Exit;

end;

end;

SetLength(a, width\_a); // Инициализация массива А и ввод его размерности

for i:=0 to width\_a-1 do

begin

SetLength(a[i], len\_a);

for j:=0 to len\_a-1 do

begin

a[i,j] := StrToFloat(Form1.matrix\_a\_sg.Cells[j, i]);

end;

end;

SetLength(b, width\_b); // Инициализация массива В и ввод его размерности

for i:=0 to width\_b-1 do

begin

SetLength(b[i], len\_b);

for j:=0 to len\_b-1 do

begin

b[i,j] := StrToFloat(Form1.matrix\_b\_sg.Cells[j, i]);

end;

end;

init := True;

end;

procedure calculate\_determinant; // Вычисление определителя матрицы A методом Гаусса

var i, j, k, r, n: Integer; s: string; max, c, m: Real; temp\_arr: arr\_type;

begin

with Form1 do

begin

if len\_a <> width\_a then

begin

det\_error\_lbl.Visible := True;

Exit;

end;

n:= len\_a-1;

SetLength(temp\_arr, n+1, n+1);

for i:=0 to n do

for j:=0 to n do

begin

temp\_arr[i,j] := a[i,j];

end;

result:=1;

for k:=0 to n do

begin

max:=abs(temp\_arr[k,k]);

r:=k;

for i:=k+1 to n do

begin

if abs(temp\_arr[i,k])>max then

begin

max:=abs(temp\_arr[i,k]);

r:=i;

end;

end;

if r<>k then result:=-result;

for j:=0 to n do

begin

c:=temp\_arr[k,j];

temp\_arr[k,j]:=temp\_arr[r,j];

temp\_arr[r,j]:=c;

end;

for i:=k+1 to n do

begin

if Abs(temp\_arr[k,k]) < 0.0001 then

begin

result := 0;

Str(result:0:3, s);

det\_a\_edt.Text := s;

det\_a\_edt.Visible := True;

Exit;

end;

m:=temp\_arr[i,k]/temp\_arr[k,k];

for j:=k to n do

temp\_arr[i,j]:=temp\_arr[i,j]-m\*temp\_arr[k,j];

end;

end;

for i:=0 to n do

result:=result\*temp\_arr[i,i];

if Abs(result) < 0.0001 then

begin

result := 0;

Str(result:0:3, s);

det\_a\_edt.Text := s;

det\_a\_edt.Visible := True;

Exit;

end;

Str(result:0:3, s);

det\_a\_edt.Text := s;

det\_a\_edt.Visible := True;

end;

end;

procedure calculate\_a\_plus\_b; // Сложение А + В

var i, j: Integer; s: string; temp: Real;

begin

with Form1 do

begin

if (len\_a + width\_a <> width\_b +len\_b) then

begin

a\_plus\_b\_sg.Visible := False;

error\_a\_plus\_b.Visible := True;

Exit;

end;

for i:=0 to width\_a-1 do

for j:=0 to len\_a-1 do

begin

temp := a[i, j] + b[i, j];

Str(temp:0:2, s);

a\_plus\_b\_sg.Cells[j, i] := s;

end;

a\_plus\_b\_sg.ColCount := len\_a;

a\_plus\_b\_sg.RowCount := width\_a;

a\_plus\_b\_sg.Visible := True;

end;

end;

procedure calculate\_a\_minus\_b; // Вычитание А - В

var i, j: Integer; s: string; temp: Real;

begin

with Form1 do

begin

if (len\_a + width\_a <> width\_b +len\_b) then

begin

a\_minus\_b\_sg.Visible := False;

error\_a\_minus\_b.Visible := True;

Exit;

end;

for i:=0 to width\_a-1 do

for j:=0 to len\_a-1 do

begin

temp := a[i, j] - b[i, j];

Str(temp:0:2, s);

a\_minus\_b\_sg.Cells[j, i] := s;

end;

a\_minus\_b\_sg.ColCount := len\_a;

a\_minus\_b\_sg.RowCount := width\_a;

a\_minus\_b\_sg.Visible := True;

end;

end;

procedure calculate\_a\_multiply\_b; // Умножение A \* B

var i, j, k: Integer; s: string;

temp\_arr: arr\_type;

c: mas;

begin

with Form1 do

begin

if not (width\_b = len\_a) then

begin

a\_mult\_b\_sg.Visible := False;

error\_a\_mult\_b.Visible := True;

Exit;

end;

SetLength(temp\_arr, width\_a, len\_b);

for i:=0 to width\_a-1 do

for j:=0 to len\_b-1 do

begin

for k:=0 to width\_b-1 do

begin

temp\_arr[i,j] := temp\_arr[i,j] + a[i, k] \* b[k, j];

end;

Str(temp\_arr[i,j]:0:2, s);

a\_mult\_b\_sg.Cells[j, i] := s;

end;

a\_mult\_b\_sg.ColCount := len\_b;

a\_mult\_b\_sg.RowCount := width\_a;

a\_mult\_b\_sg.Visible := True;

end;

end;

procedure multiply\_a\_by\_num; // Умножение матрицы А на число а

var i, j: Integer; s: string; temp: Real;

begin

with Form1 do

begin

for i:=0 to width\_a-1 do

for j:=0 to len\_a-1 do

begin

temp := a[i, j] \* nums\_trackbar.Position;

Str(temp:0:2, s);

a\_mult\_by\_num.Cells[j, i] := s;

end;

a\_mult\_by\_num.ColCount := len\_a;

a\_mult\_by\_num.RowCount := width\_a;

a\_mult\_by\_num.Visible := True;

end;

end;

procedure transpose\_a; // Транспонирование А

var i, j: Integer; s: string; temp: Real; temp\_arr: arr\_type;

begin

with Form1 do

begin

SetLength(temp\_arr, len\_a, width\_a);

for i:=0 to width\_a-1 do

for j:=0 to len\_a-1 do

begin

temp\_arr[j, i] := a[i, j];

temp := temp\_arr[j, i];

Str(temp:0:2, s);

transposed\_a\_sg.Cells[i, j] := s;

end;

transposed\_a\_sg.ColCount := width\_a;

transposed\_a\_sg.RowCount := len\_a;

transposed\_a\_sg.Visible := True;

end;

end;

procedure calculate\_inversed\_matrix\_a; // Нахождение обратной матрицы для А

var i, j, k, kol: Integer; s: string; temp: Real; temp\_arr, temp\_arr1: arr\_type;

begin

with Form1 do

begin

if (len\_a <> width\_a) or (Abs(result) < 0.001) then

begin

error\_inversed.Visible := True;

Exit;

end;

kol := len\_a;

SetLength(temp\_arr, kol, kol);

SetLength(temp\_arr1, kol, kol);

for i:=0 to kol-1 do

for j:=0 to kol-1 do

begin

temp\_arr1[i,j] := a[i,j];

end;

for k:=0 to kol - 1 do

begin

for i:=0 to kol - 1 do

for j:=0 to kol - 1 do

begin

if (i=k) and (j=k) then

temp\_arr[i,j] := 1/temp\_arr1[i,j];

if (i=k) and (j<>k) then

temp\_arr[i,j] := temp\_arr1[i,j]/temp\_arr1[k,k];

if (i<>k) and (j=k) then

temp\_arr[i,j] := -temp\_arr1[i,k]/temp\_arr1[k,k];

if (i<>k) and (j<>k) then

temp\_arr[i,j] := temp\_arr1[i,j] - temp\_arr1[k,j] \* temp\_arr1[i,k]/temp\_arr1[k,k];

end;

for i:= 0 to kol - 1 do

for j:= 0 to kol - 1 do

begin

temp\_arr1[i,j] := temp\_arr[i,j]

end;

end;

for i:=0 to kol-1 do

for j:=0 to kol-1 do

begin

temp := temp\_arr1[i, j];

Str(temp:0:4, s);

inversed\_a\_sg.Cells[j, i] := s;

end;

inversed\_a\_sg.ColCount := width\_a;

inversed\_a\_sg.RowCount := len\_a;

inversed\_a\_sg.Visible := True;

end;

end;

procedure usual\_sort; // Сортировка матрицы А

var i, j, n, j\_min: integer;

min:Real;

s: string;

temp\_row, ans\_row: array of Real;

begin

with Form1 do

begin

sorted\_a\_sg.Visible := True;

sorted\_a\_sg.RowCount := width\_a;

sorted\_a\_sg.ColCount := len\_a;

for n := 0 to width\_a - 1 do

begin

SetLength(temp\_row, len\_a);

SetLength(ans\_row, len\_a);

for j := 0 to len\_a - 1 do

begin

ans\_row[j] := a[n, j];

end;

for j := 0 to len\_a - 1 do

begin

min := ans\_row[j];

j\_min := j;

for i := 0 to len\_a - 1 do

begin

if ans\_row[i] < min then

begin

min := ans\_row[i];

j\_min := i;

end;

end;

temp\_row[j] := min;

ans\_row[j\_min] := 9999;

end;

for j := 0 to len\_a - 1 do

begin

Str(temp\_row[j]:0:2, s);

Form1.sorted\_a\_sg.Cells[j, n] := s;

end;

end;

end;

end;

{

procedure sort;

var i,j,k,m: Integer;

nech,ch,temp: Extended;

s: string;

begin

if width\_a<>len\_a then

begin

Form1.lbl2.Visible:=True;;

Exit;

end;

Form1.SG.Visible:=True;

Form1.SG.RowCount := width\_a;

Form1.SG.ColCount := len\_a;

nech:=0;

ch:=0;

m:=0;

{ for i:=1 to width\_a do

begin

inc(m);

if (m mod 2 = 0) then

ch:=ch+a[i,i];

if (m mod 2 <> 0) then

nech:=nech+a[i,i];

end;}

{for i:=0 to width\_a-1 do

begin

if (i mod 2 = 0) and (i > 0) and (i <= len\_a) then

ch:=ch+a[i,i]

else

nech:=nech+a[i,i];

end;

if ch<nech then

for i:=0 to width\_a-1 do

for j:=0 to len\_a-1 do

for k:=0 to len\_a-2 do

if a[i,k]<a[i,k+1] then

begin

temp:=a[i,k];

a[i,k]:=a[i,k+1];

a[i,k+1]:=temp;

end;

for i:=0 to width\_a-1 do

for j:=0 to len\_a-1 do

begin

temp := a[i, j];

Str(temp:0:2, s);

Form1.SG.Cells[j, i] := s;

end;

end;

}

procedure TForm1.up\_down\_len\_aClick(Sender: TObject; Button: TUDBtnType);

begin

len\_a := StrToInt(len\_of\_arr\_a.Text);

matrix\_a\_sg.ColCount := len\_a;

FormClick(Form1);

end;

procedure TForm1.up\_down\_width\_aClick(Sender: TObject; Button: TUDBtnType);

begin

width\_a := StrToInt(width\_of\_arr\_a.Text);

matrix\_a\_sg.RowCount := width\_a;

FormClick(Form1);

end;

procedure TForm1.up\_down\_len\_bClick(Sender: TObject; Button: TUDBtnType);

begin

len\_b := StrToInt(len\_of\_arr\_b.Text);

matrix\_b\_sg.ColCount := len\_b;

FormClick(Form1);

end;

procedure TForm1.up\_down\_width\_bClick(Sender: TObject; Button: TUDBtnType);

begin

width\_b := StrToInt(width\_of\_arr\_b.Text);

matrix\_b\_sg.RowCount := width\_b;

FormClick(Form1);

end;

procedure TForm1.close\_btnClick(Sender: TObject);

begin

Form1.Close;

Form2.Close;

end;

procedure TForm1.calc\_btnClick(Sender: TObject);

begin

if init() then

begin

a\_minus\_b\_sg.Visible := False;

a\_mult\_by\_num.Visible := False;

a\_minus\_b\_sg.Visible := False;

a\_plus\_b\_sg.Visible := False;

error\_a\_minus\_b.Visible := False;

error\_a\_plus\_b.Visible := False;

error\_a\_mult\_b.Visible := False;

det\_error\_lbl.Visible := False;

det\_a\_edt.Visible := False;

inversed\_a\_sg.Visible := False;

error\_inversed.Visible := False;

multiply\_a\_by\_num;

calculate\_determinant;

transpose\_a;

calculate\_a\_plus\_b;

calculate\_a\_minus\_b;

calculate\_a\_multiply\_b;

calculate\_inversed\_matrix\_a;

usual\_sort;

page.Visible := True;

matrix\_a\_lbl.Visible := True;

matrix\_b\_sg.Visible := True;

end;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

len\_a := 1;

len\_b := 1;

width\_a := 1;

width\_b := 1;

end;

procedure TForm1.read\_from\_file\_btnClick(Sender: TObject);

var F:TextFile; k: string; i, j: integer;

begin

if dlgOpen.Execute then

begin

AssignFile(F, dlgOpen.FileName);

Reset(F);

Readln(F, k);

len\_a := StrToInt(k);

up\_down\_len\_a.Position := len\_a;

Readln(F, k);

width\_a := StrToInt(k);

up\_down\_width\_a.Position := width\_a;

Readln(F, k);

len\_b := StrToInt(k);

up\_down\_len\_b.Position := len\_b;

Readln(F, k);

width\_b := StrToInt(k);

up\_down\_width\_b.Position := width\_b;

for i:=0 to len\_a-1 do

for j:=0 to width\_a-1 do

begin

Readln(F, k);

matrix\_a\_sg.Cells[i, j] := k;

end;

for i:=0 to len\_b-1 do

for j:=0 to width\_b-1 do

begin

Readln(F, k);

matrix\_b\_sg.Cells[i, j] := k;

end;

matrix\_a\_sg.RowCount := width\_a;

matrix\_a\_sg.ColCount := len\_a;

matrix\_b\_sg.RowCount := width\_b;

matrix\_b\_sg.ColCount := len\_b;

CloseFile(F);

enter\_values\_btnClick(Form1);

FormClick(Form1);

matrix\_a\_sg.Visible := True;

matrix\_b\_sg.Visible := True;

end;

end;

procedure TForm1.write\_to\_file\_btnClick(Sender: TObject);

var F: TextFile; i, j: integer;

begin

if dlgSave.Execute then

begin

AssignFile(F, dlgSave.FileName);

Rewrite(F);

Writeln(F, inttostr(len\_a));

Writeln(F, inttostr(width\_a));

Writeln(F, inttostr(len\_b));

Writeln(F, inttostr(width\_b));

for i:=0 to len\_a-1 do

for j:=0 to width\_a-1 do

begin

Writeln(F, matrix\_a\_sg.Cells[i, j]);

end;

for i:=0 to len\_b-1 do

for j:=0 to width\_b-1 do

begin

Writeln(F, matrix\_b\_sg.Cells[i, j]);

end;

CloseFile(F);

end;

matrix\_a\_sg.Visible := True;

matrix\_b\_sg.Visible := True;

calc\_btn.Visible := True;

end;

procedure TForm1.enter\_values\_btnClick(Sender: TObject);

begin

grp2.Visible:=True;

end;

procedure TForm1.FormClick(Sender: TObject);

begin

grp1.Visible:=True;

end;

procedure TForm1.matrix\_a\_sgKeyPress(Sender: TObject; var Key: Char);

begin

if Key = '.' then Key := ',';

if not (Key in ['0'..'9', ',', '.', '-', #08]) then Key:= #0;

end;

procedure TForm1.matrix\_b\_sgKeyPress(Sender: TObject; var Key: Char);

begin

if Key = '.' then Key := ',';

if not (Key in ['0'..'9', ',', '.', '-', #08]) then Key:= #0;

end;

procedure TForm1.bgClick(Sender: TObject);

begin

matrix\_a\_lbl.Visible:=True;

matrix\_a\_sg.Visible:=True;

matrix\_b\_lbl.Visible:=True;

matrix\_b\_sg.Visible:=True;

end;

end.;

ПРИЛОЖЕНИЕ В

**Код программы для обработки строк**

**unit** Unit1unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls;

type

TForm1 = class(TForm)

m1: TMemo;

m2: TMemo;

cb1: TComboBox;

rg1: TRadioGroup;

txt1: TStaticText;

rb1: TRadioButton;

rb2: TRadioButton;

l1: TLabel;

l2: TLabel;

l3: TLabel;

procedure FormActivate(Sender: TObject);

procedure cb1Change(Sender: TObject);

procedure rb1Click(Sender: TObject);

procedure rb2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

type

astr = array[1..128] of string;

var

Form1: TForm1;

implementation

{$R \*.dfm}

procedure TForm1.FormActivate(Sender: TObject);

begin

m1.SetFocus;

end;

procedure TForm1.cb1Change(Sender: TObject);

begin

if cb1.ItemIndex=0

then begin

txt1.Visible:=true;

txt1.Caption:='Дан текст. Все слова, длина которых равна длине слова максимальной длины, заменить на слово максимальной длины.';

end;

if cb1.ItemIndex=1

then begin

txt1.Visible:=true;

txt1.Caption:='Дан текст. Слова в тексте разделены пробелами. Вычеркнуть из текста все повторяющиеся слова.';

end;

end;

procedure TForm1.rb1Click(Sender: TObject);

var

text: string;

words: array[1..100] of string; // Массив для хранения слов

maxLength, i, wordCount: Integer;

maxLengthWord: string;

currentWord: string;

ch: Char;

result: string;

begin

Text:=m1.Text;

wordCount := 0;

currentWord := '';

for i := 1 to Length(text) do

begin

ch := text[i];

if (ch <> ' ') and (ch <> '.') and (ch <> ',') and (ch <> ';') and (ch <> '!') and (ch <> '?') then

currentWord := currentWord + ch

else

begin

if currentWord <> '' then

begin

wordCount := wordCount + 1;

words[wordCount] := currentWord;

currentWord := '';

end;

end;

end;

// Добавляем последнее слово, если оно есть

if currentWord <> '' then

begin

wordCount := wordCount + 1;

words[wordCount] := currentWord;

end;

// Находим максимальную длину слова

maxLength := 0;

for i := 1 to wordCount do

begin

if Length(words[i]) > maxLength then

begin

maxLength := Length(words[i]);

maxLengthWord := words[i];

end;

end;

// Заменяем слова максимальной длины

for i := 1 to wordCount do

begin

if Length(words[i]) = maxLength then

words[i] := maxLengthWord;

end;

// Выводим измененный текст

for i := 1 to wordCount do

begin

result:=result+words[i]+' ';

end;

m2.Text:=result;

txt1.Visible:=true;

txt1.Caption:='1) Дан текст. Все слова, длина которых равна длине слова максимальной длины, заменить на слово максимальной длины.';

cb1.ItemIndex:=0;

end;

procedure TForm1.rb2Click(Sender: TObject);

var

s, result, word: string;

i,j: integer;

words: array of string;

found: boolean;

Begin

s:=m1.Text;

result := ''; // Итоговая строка

SetLength(words, 0); // Массив для хранения уникальных слов

i := 1; // Инициализация индекса

while i <= length(s) do

begin

if s[i] in ['.', ',', '!', '?', ';', ':', '-', '(', ')', '"', ''''] then

begin

// Если встречаем знак препинания, добавляем его в результат

if result <> '' then

result := result + ' '; // Добавляем пробел перед знаком

result := result + s[i]; // Добавляем знак препинания

i := i + 1; // Переходим к следующему символу

end

else if s[i] <> ' ' then

begin

// Формируем слово

word := '';

while (i <= length(s)) and (s[i] <> ' ') and not (s[i] in ['.', ',', '!', '?', ';', ':', '-', '(', ')', '"', '''']) do

begin

word := word + s[i];

i := i + 1;

end;

// Проверяем, есть ли такое слово в массиве

found := false;

for j := 0 to High(words) do

begin

if words[j] = word then

begin

found := true;

break;

end;

end;

// Если слово уникально, добавляем его в результат и массив

if not found then

begin

if result <> '' then

result := result + ' '; // Добавляем пробел перед новым словом

result := result + word; // Добавляем слово в результат

// Добавляем слово в массив

SetLength(words, Length(words) + 1);

words[High(words)] := word;

end;

end

else

i := i + 1; // Пропускаем пробелы

end;

m2.Text:=result;

txt1.Visible:=true;

txt1.Caption:='2) Дан текст. Слова в тексте разделены пробелами. Вычеркнуть из текста все повторяющиеся слова.';

cb1.ItemIndex:=1;

End;

end.

# ПРИЛОЖЕНИЕ Г

**Код программы для обработки СЛАУ**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Grids, StdCtrls, Buttons, ComCtrls;

type

TForm1 = class(TForm)

lbl1: TLabel;

SG1: TStringGrid;

lbl2: TLabel;

SG2: TStringGrid;

lbl3: TLabel;

edt1: TEdit;

edt2: TEdit;

ud1: TUpDown;

ud2: TUpDown;

CBB1: TComboBox;

lbl4: TLabel;

btn1: TBitBtn;

btn2: TBitBtn;

btn3: TBitBtn;

btn4: TBitBtn;

btn5: TBitBtn;

pgc1: TPageControl;

tsblock: TTabSheet;

tskram: TTabSheet;

tsprov: TTabSheet;

SGBl: TStringGrid;

SGKr: TStringGrid;

SGPr: TStringGrid;

OpenDialog1: TOpenDialog;

SaveDialog1: TSaveDialog;

procedure SG1KeyPress(Sender: TObject; var Key: Char);

procedure SG2KeyPress(Sender: TObject; var Key: Char);

procedure Float1(var Key: char);

procedure Float2(var Key: char);

procedure btn2Click(Sender: TObject);

procedure FormClick(Sender: TObject);

procedure btn1Click(Sender: TObject);

procedure btn3Click(Sender: TObject);

procedure btn4Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

type

dvmas = array of array of double;

odmas = array of double;

var

Form1: TForm1;

i: integer;

implementation

function error():Boolean;

var

m,n,j:integer;

begin

m:=strtoint(form1.edt1.text);

n:=strtoint(form1.edt2.text);

Result:=False;

for i:=1 to m do

for j:=1 to n do

if form1.SG1.Cells[j,i]='' then

begin

Messagedlg('Матрица A не заполненна!',mterror,[mbok],0);

Result:=True;

exit;

end;

for i:=1 to m do

if form1.SG2.Cells[1,i]='' then

begin

Messagedlg('Вектор B не заполнен!',mterror,[mbok],0);

Result:=True;

exit;

end;

end;

{$R \*.dfm}

procedure createMatrix(var a: dvmas);

var

m, n, i, j: Integer;

begin

m:=StrToInt(Form1.edt1.text);

n:=StrToInt(Form1.edt2.text);

SetLength(a, m, n);

for i:=1 to m do

for j:=1 to n do

begin

a[i-1,j-1]:=StrToFloat(Form1.SG1.Cells[j,i]);

end;

end;

procedure createVektor(var b: odmas);

var

n, i: Integer;

begin

n:=StrToInt(Form1.edt1.text);

SetLength(b, n);

for i:=1 to n do

b[i-1]:=StrToFloat(Form1.SG2.Cells[1,i]);

end;

function validateSystem(): Integer;

var

m, n: Integer;

begin

m:=StrToInt(Form1.edt1.text);

n:=StrToInt(Form1.edt2.text);

if(m=n) then

validateSystem:=1

else

begin

MessageDlg('СЛАУ не квадратная!',mtError,[mbOK],0);

validateSystem:=0;

exit;

end;

end;

function pr(a: dvmas): odmas;

var

n, i, k: Integer;

b, x: odmas;

sum: double;

begin

n:=StrToInt(Form1.edt1.text);

SetLength(b,n);

createVektor(b);

SetLength(x,n);

For i:=0 to n-1 do

Begin

sum:=0;

For k:=0 to n-1 do

sum:=sum+a[i,k]\*b[k];

x[i]:=sum;

End;

pr:=x;

end;

function reverse(a: dvmas): dvmas;

var

n, i, j, k: Integer;

ar, p: dvmas;

begin

n:=StrToInt(Form1.edt1.text);

SetLength(ar, n, n);

SetLength(p, n, n);

for i:=0 to n-1 do

for j:=0 to n-1 do

ar[i,j]:=a[i,j];

for k:=0 to n-1 do

begin

for i:=0 to n-1 do

for j:=0 to n-1 do

begin

if(ar[k,k]<>0) then

begin

if((i<>k)and(j<>k))then

p[i,j]:=ar[i,j]-ar[k,j]\*ar[i,k]/ar[k,k];

if((i=k)and(j<>k))then

p[i,j]:=ar[k,j]/ar[k,k];

if((i<>k)and(j=k))then

p[i,j]:=-ar[i,k]/ar[k,k];

if((i=k)and(j=k))then

p[i,j]:=1/ar[k,k];

end;

end;

for i:=0 to n-1 do

for j:=0 to n-1 do

ar[i,j]:=p[i,j];

end;

reverse:=ar;

end;

function det(a: dvmas): Double;

var

m, n, i, j, k: Integer;

p: Double;

begin

m:=StrToInt(Form1.edt1.text);

n:=StrToInt(Form1.edt2.text);

if(m=n) then

begin

p:=1;

for k:=0 to n-1 do

begin

if(a[k,k]=0) then

begin

det:=0;

exit;

end;

for i:=k+1 to n-1 do

begin

for j:=k+1 to n-1 do

a[i,j]:=a[i,j]-a[k,j]/a[k,k]\*a[i,k];

end;

end;

for i:=0 to n-1 do

p:=p\*a[i,i];

det:=p;

end;

end;

function checkDet(): Integer;

var

m, n: Integer;

a: dvmas;

begin

m:=StrToInt(Form1.edt1.text);

n:=StrToInt(Form1.edt2.text);

SetLength(a,n,m);

createMatrix(a);

if ((m=n) and (det(a)=0)) then

begin

MessageDlg('Определитель исходной матрицы равен 0 или СЛАУ имеет бесконечно много решений!', mtError, [mbOK], 0);

checkDet:=0

end

else

checkDet:=1;

end;

procedure showBlock(x: odmas);

var

m, i, j: Integer;

begin

m:=StrToInt(Form1.edt2.text);

Form1.SGBl.RowCount:=m+1;

for i:=0 to m-1 do

Form1.SGBl.Cells[1,i+1]:=FloatToStrF(x[i], ffFixed, 3, 3);

form1.SGBl.Visible:=true;

form1.pgc1.Visible:=true;

end;

procedure block();

var

m, n: Integer;

a: dvmas;

x: odmas;

begin

m:=StrToInt(Form1.edt1.Text);

n:=StrToInt(Form1.edt2.text);

SetLength(a,m,n);

createMatrix(a);

a:=reverse(a);

x:=pr(a);

showBlock(x);

end;

procedure kram();

var

m, n, i, j, temp: integer;

tempV, deter: double;

a: dvmas;

b, x: odmas;

begin

m:=StrToInt(Form1.edt1.text);

n:=StrToInt(Form1.edt2.text);

SetLength(a,m,n);

createMatrix(a);

SetLength(b,n);

createVektor(b);

SetLength(x,m);

tempV:=det(a);

if (tempV=0) then

begin

Form1.SGKr.Visible:=False;

exit;

end

else

Form1.SGKr.Visible:=True;

Form1.SGKr.Visible:=True;

Form1.pgc1.Visible:=True;

for i:=0 to n-1 do

begin

createMatrix(a);

for j:=0 to n-1 do

begin

a[j,i]:=b[j];

end;

deter:=det(a);

if(tempV<>0) then

x[i]:=deter/tempV

else

x[i]:=0;

end;

temp:=1;

Form1.SGKr.RowCount:=m+1;

for i:=0 to m-1 do

Form1.SGKr.Cells[1,i+1]:=FloatToStrF(x[i],ffFixed,3,3);

createMatrix(a);

tempV:=det(a);

end;

procedure TForm1.Float1(var Key: char);

var

allow: string;

Row: integer;

ok: boolean;

begin

allow := '1234567890-,' + #8;

ok := false;

// Проверка на ввод нуля в начале строки

if (Key = '0') and (Length(SG1.Cells[SG1.Col, SG1.Row]) = 1) and

(Pos('0', SG1.Cells[SG1.Col, SG1.Row]) = 1) then

ok := false

// Проверка на ввод запятой, если строка пуста или уже есть запятая

else if (Key = ',') and ((Length(SG1.Cells[SG1.Col, SG1.Row]) = 0) or

(SG1.Cells[SG1.Col, SG1.Row][Length(SG1.Cells[SG1.Col, SG1.Row])] = ',') or

(Pos(',', SG1.Cells[SG1.Col, SG1.Row]) > 0)) then

ok := false

// Проверка на ввод минуса, если строка не пуста или уже есть минус

else if (Key = '-') and ((Length(SG1.Cells[SG1.Col, SG1.Row]) > 0) or

(Pos('-', SG1.Cells[SG1.Col, SG1.Row]) > 1)) then

ok := false

// Проверка на ввод запятой после минуса, если строка состоит только из минуса

else if (Key = ',') and (Pos('-', SG1.Cells[SG1.Col, SG1.Row]) = 1) and

(Length(SG1.Cells[SG1.Col, SG1.Row]) = 1) then

ok := false

else

begin

// Проверка на допустимые символы

for Row := 1 to Length(allow) do

if Key = allow[Row] then

ok := true;

end;

if not ok then

Key := #0;

end;

procedure TForm1.Float2(var Key: char);

var

allow: string;

Row: integer;

ok: boolean;

begin

allow := '1234567890-,' + #8;

ok := false;

// Проверка на ввод нуля в начале строки

if (Key = '0') and (Length(SG2.Cells[SG2.Col, SG2.Row]) = 1) and

(Pos('0', SG2.Cells[SG2.Col, SG2.Row]) = 1) then

ok := false

// Проверка на ввод запятой, если строка пуста или уже есть запятая

else if (Key = ',') and ((Length(SG2.Cells[SG2.Col, SG2.Row]) = 0) or

(SG2.Cells[SG2.Col, SG2.Row][Length(SG2.Cells[SG2.Col, SG2.Row])] = ',') or

(Pos(',', SG2.Cells[SG2.Col, SG2.Row]) > 0)) then

ok := false

// Проверка на ввод минуса, если строка не пуста или уже есть минус

else if (Key = '-') and ((Length(SG2.Cells[SG2.Col, SG2.Row]) > 0) or

(Pos('-', SG2.Cells[SG2.Col, SG2.Row]) > 1)) then

ok := false

// Проверка на ввод запятой после минуса, если строка состоит только из минуса

else if (Key = ',') and (Pos('-', SG2.Cells[SG2.Col, SG2.Row]) = 1) and

(Length(SG2.Cells[SG2.Col, SG2.Row]) = 1) then

ok := false

else

begin

// Проверка на допустимые символы

for Row := 1 to Length(allow) do

if Key = allow[Row] then

ok := true;

end;

if not ok then

Key := #0;

end;

procedure TForm1.SG1KeyPress(Sender: TObject; var Key: Char);

begin

Float1(Key);

end;

procedure TForm1.SG2KeyPress(Sender: TObject; var Key: Char);

begin

Float2(Key);

end;

procedure TForm1.btn2Click(Sender: TObject);

begin

lbl3.Visible:=true;

edt1.Visible:=true;

ud1.Visible:=true;

edt2.Visible:=true;

ud2.Visible:=True;

end;

procedure TForm1.FormClick(Sender: TObject);

var i: Integer;

begin

Form1.SG1.RowCount:=StrToInt(edt1.Text)+1;

Form1.SG1.ColCount:=StrToInt(edt2.Text)+1;

Form1.SG2.RowCount:=StrToInt(edt1.Text)+1;

lbl1.Visible:=true;

sg1.Visible:=true;

lbl2.Visible:=true;

sg2.visible:=true;

lbl4.visible:=true;

cbb1.Visible:=true;

end;

procedure checkedKr();

var

m, n, i, j: integer;

ax, x: odmas;

a: dvmas;

sum: double;

begin

m:=StrToInt(Form1.edt1.text);

n:=StrToInt(Form1.edt2.text);

SetLength(a, m, n);

SetLength(x, m);

SetLength(ax, m);

for i:=0 to m-1 do

for j:=0 to n-1 do

a[i,j]:=StrToFloat(Form1.SG1.Cells[j+1,i+1]);

for i:=0 to m-1 do

x[i]:=StrToFloat(Form1.SGKr.Cells[1,i+1]);

Form1.SGPr.RowCount:=m+1;

Form1.SGPr.ColCount:=2;

for i:=0 to m-1 do

begin

sum:=0;

for j:=0 to n-1 do

sum:=sum+a[i,j]\*x[j];

ax[i]:=sum;

Form1.SGPr.Cells[1,i+1]:=FloatToStrF(ax[i],ffFixed,3,3);

end;

Form1.SGPr.Visible:=true;

end;

procedure checkedBl();

var

m, n, i, j: integer;

ax, x: odmas;

a: dvmas;

sum: double;

begin

m:=StrToInt(Form1.edt1.text);

n:=StrToInt(Form1.edt2.text);

SetLength(a, m, n);

SetLength(x, m);

SetLength(ax, m);

for i:=0 to m-1 do

for j:=0 to n-1 do

a[i,j]:=StrToFloat(Form1.SG1.Cells[j+1,i+1]);

for i:=0 to m-1 do

x[i]:=StrToFloat(Form1.SGBl.Cells[1,i+1]);

Form1.SGPr.RowCount:=m+1;

Form1.SGPr.ColCount:=2;

for i:=0 to m-1 do

begin

sum:=0;

for j:=0 to n-1 do

sum:=sum+a[i,j]\*x[j];

ax[i]:=sum;

Form1.SGPr.Cells[1,i+1]:=FloatToStrF(ax[i],ffFixed,3,3);

end;

Form1.SGPr.Visible:=true;

end;

procedure TForm1.btn1Click(Sender: TObject);

var

m, n, i, j: integer;

b, x, ax: odmas;

a: dvmas;

sum: double;

begin

if error() then Exit;

if(checkDet=0) then

exit;

if (validateSystem=0) then

exit;

if(cbb1.ItemIndex=0) then

begin

pgc1.Visible:=true;

block;

checkedBl;

pgc1.ActivePageIndex:=0;

SGKr.Visible:=false;

end

else

if (cbb1.ItemIndex=1) then

begin

pgc1.Visible:=true;

kram;

checkedKr;

pgc1.ActivePageIndex:=1;

SGBl.Visible:=false;

end

else

MessageDlg('Выберите метод решения!',mtError,[mbOK],0);

end;

procedure TForm1.btn3Click(Sender: TObject);

var

m, n, i, j: integer;

f: TextFile;

s: string;

a: dvmas;

b: odmas;

begin

OpenDialog1.Filter:='Текстовые файлы|\*.txt|';

if OpenDialog1.Execute and FileExists(OpenDialog1.Filename) then

begin

AssignFile (f,OpenDialog1.Filename);

Reset(f);

readln(f,m);

readln(f,n);

SetLength(a,m,n);

for i:=1 to m do

for j:=1 to n do

begin

readln (f,s);

SG1.Cells[j,i]:=s;

end;

readln(f,n);

SetLength(b,n);

for i:=1 to m do

for j:=1 to n do

begin

readln (f,s);

SG2.Cells[i,j]:=s;

end;

CloseFile (f);

end;

Form1.edt1.Text:=IntToStr(m);

Form1.edt2.Text:=IntToStr(n);

Form1.SG2.RowCount:=n+1;

Form1.SG1.RowCount:=n+1;

Form1.SG1.ColCount:=n+1;

Form1.ud1.Position:=m;

Form1.ud1.Position:=n;

Form1.lbl1.Visible:=true;

Form1.lbl2.Visible:=true;

Form1.lbl3.Visible:=true;

Form1.lbl4.Visible:=true;

Form1.SG1.Visible:=true;

Form1.SG2.Visible:=true;

Form1.edt1.Visible:=true;

Form1.edt2.Visible:=true;

Form1.ud1.Visible:=true;

Form1.ud2.Visible:=true;

Form1.cbb1.Visible:=true;

end;

procedure TForm1.btn4Click(Sender: TObject);

var

m, n, i, j: integer;

f: TextFile;

a: dvmas;

b: odmas;

begin

SaveDialog1.Filter:='Текстовые файлы|\*.txt|';

m:=StrToInt(Form1.edt1.text);

n:=StrToInt(Form1.edt2.text);

if SaveDialog1.Execute then

begin

AssignFile (f,SaveDialog1.Filename);

Rewrite(f);

writeln (f,n);

writeln (f,m);

SetLength(a,n,m);

createMatrix(a);

SetLength(b,n);

createVektor(b);

for i:=0 to n-1 do

begin

for j:=0 to m-1 do

writeln (f,FloatToStr(a[i,j]));

end;

writeln (f,n);

for i:=0 to n-1 do

writeln (f,FloatToStr(b[i]));

end;

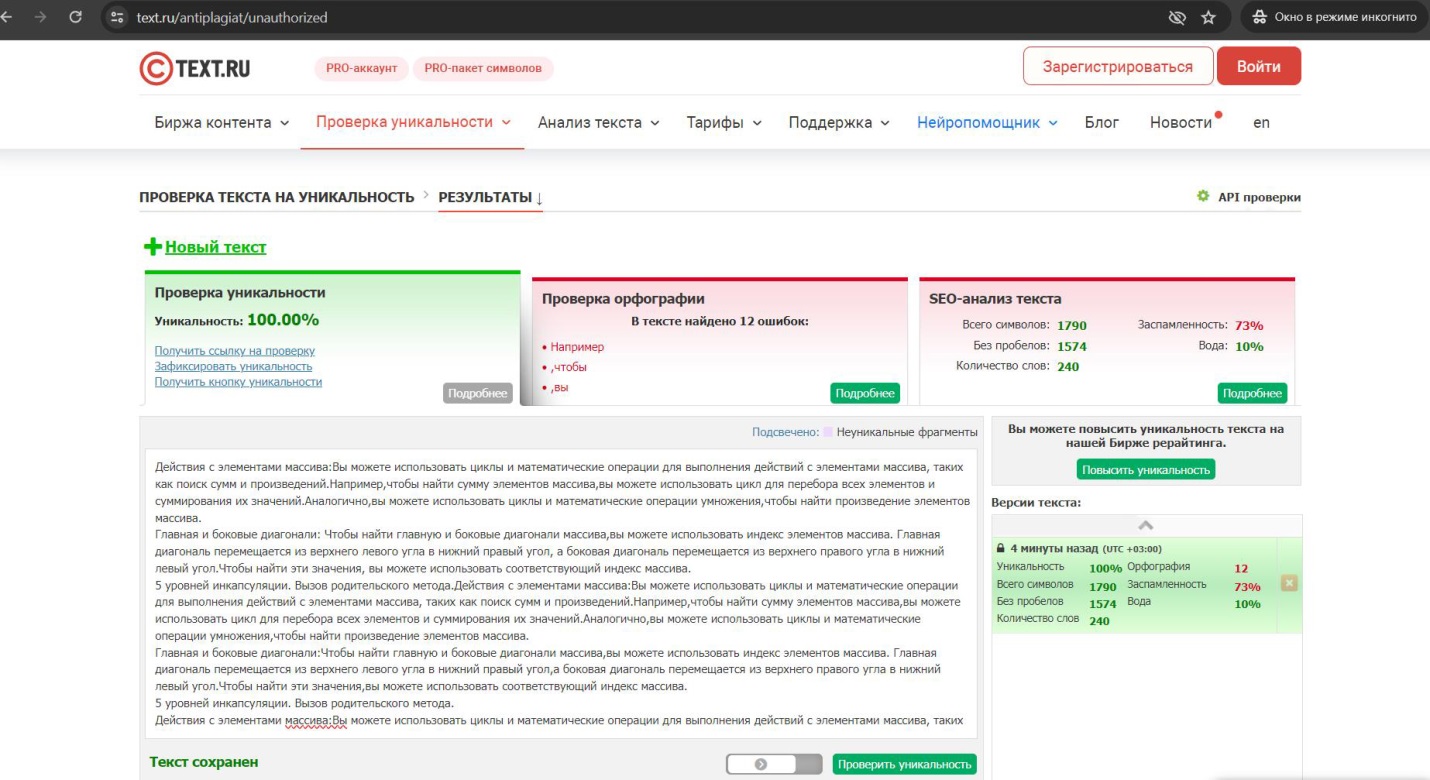
CloseFile (f);

end;

end.

# ПРИЛОЖЕНИЕ Д

**Результат проверки курсовой работы на антиплагиат**

****