Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему:

«АНАЛИЗ АЛГОРИТМОВ СЖАТИЯ ФАЙЛОВ»

БГУИР КП 1-40 01 01  013  ПЗ

Студент: гр. 851002 Ковалевский М.Ю.

Руководитель: асс. Фадеева Е.Е.

Минск 2019

**СОДЕРЖАНИЕ**

Введение ........................................................................................................ 5

1 Анализ современных алгоритмов сжатия данных........................... 7

1.1 Анализ актуальности сжатия................................... 7

1.2 Бесплатные программы для индивидуального пользования.......... 7

1.3 Условно бесплатные программы ....................................................... 10

1.4 Формирование требований к проектируемому программному

средству......................................................................................................... 11

2 Разработка алгоритм.................................................................................... 13

2.1 Анализ требований к программному средству и разработка

функциональных требований...................................................................... 13

2.2 Разработка алгоритма программного средства.................................. 14

2.3 Разработка алгоритма выполнения операций над матрицами...........16

2.4 Разработка алгоритма открытия данных из файла и сохранения

результатов в файл вручную........................................................................17

2.5 Разработка алгоритма добавления и удаления результатов

работы программного калькулятора........................................................... 18

3 Разработка программного средства.............................................................19

3.1 Разработка используемых данных....................................................... 19

3.2 Разработка схемы работы системы...................................................... 21

4 Обоснование технических приемов программирования ..........................23

5 Тестирование ...................................................... .........................................25

5.1 Тестирование математической составляющей функционала

программы......................................................................................................25

5.2 Тестирование прочих функций программы.........................................31

6 Руководство пользователя............................................................................36

Заключение.........................................................................................................42

Список использованных источников .............................................................43

Приложение. Исходный код программы .......................................................44

**ВВЕДЕНИЕ**

Самому первому компьютеру уже более 80 лет, но по сравнению даже с самым простым современным телефоном он не умел ничего. Он тратил на простые математические операции по несколько секунд и всю информацию получал с перфорированных лент, не имел собственной оперативной и встроенной памяти. Сегодня мы в мгновение ока обмениваемся сообщениями и даже мультимедийными файлами через крошечное устройство - смартфон, который не привязан ни к чему.

Мы даже не задумываемся, какой объем данных генерируем в течение дня, и где все эти данные хранятся. Благодаря интернету, Google в день обрабатывает больше данных, чем было написано во всех литературных работах на всех языках до появления интернета, и это при том, что к паутине подключена далеко не вся планета.

Безусловно, стремительный рост данных подтолкнул к появлению сайтов и распространению интернета. Первый сайт появился в 1990 году и содержал исключительно текстовую информацию - описание технологии World Wide Web.

Первый жесткий диск на 5 МБ появился 60 лет назад. Он [весил](http://gagadget.com/15739-evolyutsiya-kompyuternyih-nositelej-informatsii/) около тонны и по размеру был сравним с крупным современным холодильником. Внутри массивного корпуса находилось 50 дисков диаметром 60 сантиметров или примерно 23 дюйма. Сегодня на таком пространстве помещаются две серверные стойки, а маленькое устройство в кармане может хранить несколько сотен гигабайт информации.

Сегодня люди генерируют огромное количество данных. Если учесть, что эти данные затем копируются в облако, это только увеличивает объем памяти, необходимый для их хранения. Такие решения и технологии, как автономные автомобили, интеллектуальные фабрики, интернет вещей (IoT), блокчейн и домашняя автоматизация, будут генерировать дополнительные потоки данных, которые необходимо хранить.

Таким образом ценность сжатия данных ясна. Без уменьшения размера файлов большим компаниям, как и совершенно обычным людям, потребуется значительно больше места на накопителе. А это привело бы к дополнительным тратам на оборудование и его более быстрому износу.

Сжатие данных – это алгоритмическое преобразование [данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5), производимое с целью уменьшения занимаемого ими объёма. Применяется для более рационального использования устройств [хранения](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8) и [передачи](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) данных.

Все методы сжатия данных делятся на два основных класса:

* [Сжатие без потерь](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8C)
* [Сжатие с потерями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D1%81_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8F%D0%BC%D0%B8)

При использовании сжатия без потерь возможно полное восстановление исходных данных, сжатие с потерями позволяет восстановить данные с искажениями.

Сжатие без потерь обычно используется для передачи и хранения текстовых данных, компьютерных программ, реже - для сокращения объёма [аудио](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D0%B4%D0%B8%D0%BE)- и [видеоданных](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE), [цифровых фотографий](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D1%84%D0%BE%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) и т. п., в случаях, когда искажения недопустимы или нежелательны. Сжатие с потерями, обладающее значительно большей, чем сжатие без потерь, эффективностью, обычно применяется для сокращения объёма аудио- и видеоданных и цифровых фотографий в тех случаях, когда такое сокращение является приоритетным, а полное соответствие исходных и восстановленных данных не требуется.

Данная записка содержит следующие разделы курсовой работы по разработке ПО для проведения операций над матрицами:

1. Анализ современных программ, дающих возможность сжимать данные;
2. Разработка алгоритма на основе сформированных в результате анализа требований для программы;
3. Разработка программного средства. Выбор структур данных для использования;
4. Обоснование технических приемов программирования. Их преимущества и выгодность использования;
5. Тестирование, анализ работы программы. Описание действий, необходимых для полного тестирования программы;
6. Руководство пользователю. Включает в себя описание действий, которые позволят успешно использовать приложение.

**1 АНАЛИЗ СОВРЕМЕННЫХ ПРОГРАММ ДЛЯ СЖАТИЯ ДАННЫХ**

**1.1 Анализ актуальности сжатия**

В наше время сжатие данных используется повсеместно. Такие программы, как архиваторы, установлены на каждом компьютере. С Windows 10 в комплекте идет алгоритм сжатия LZX, доступный через PowerShell.

Знакомство со сжатыми файлами начинается с обычных картинок. Большинство изображений хранится в таком формате, как JPEG. Файлы такого формата могут иметь расширения .jpg, .jfif, .jpe или .jpeg. Однако из них .jpg является самым популярным на всех платформах.

Каждый день ведутся разработки новых алгоритмов, позволяющих сжать файл с меньшими потерями и с меньшим итоговым размером.

Сжатие данных получило широкое распространение вместе с интернетом и после изобретения алгоритмов Лемпелем и Зивом (алгоритмы LZ).

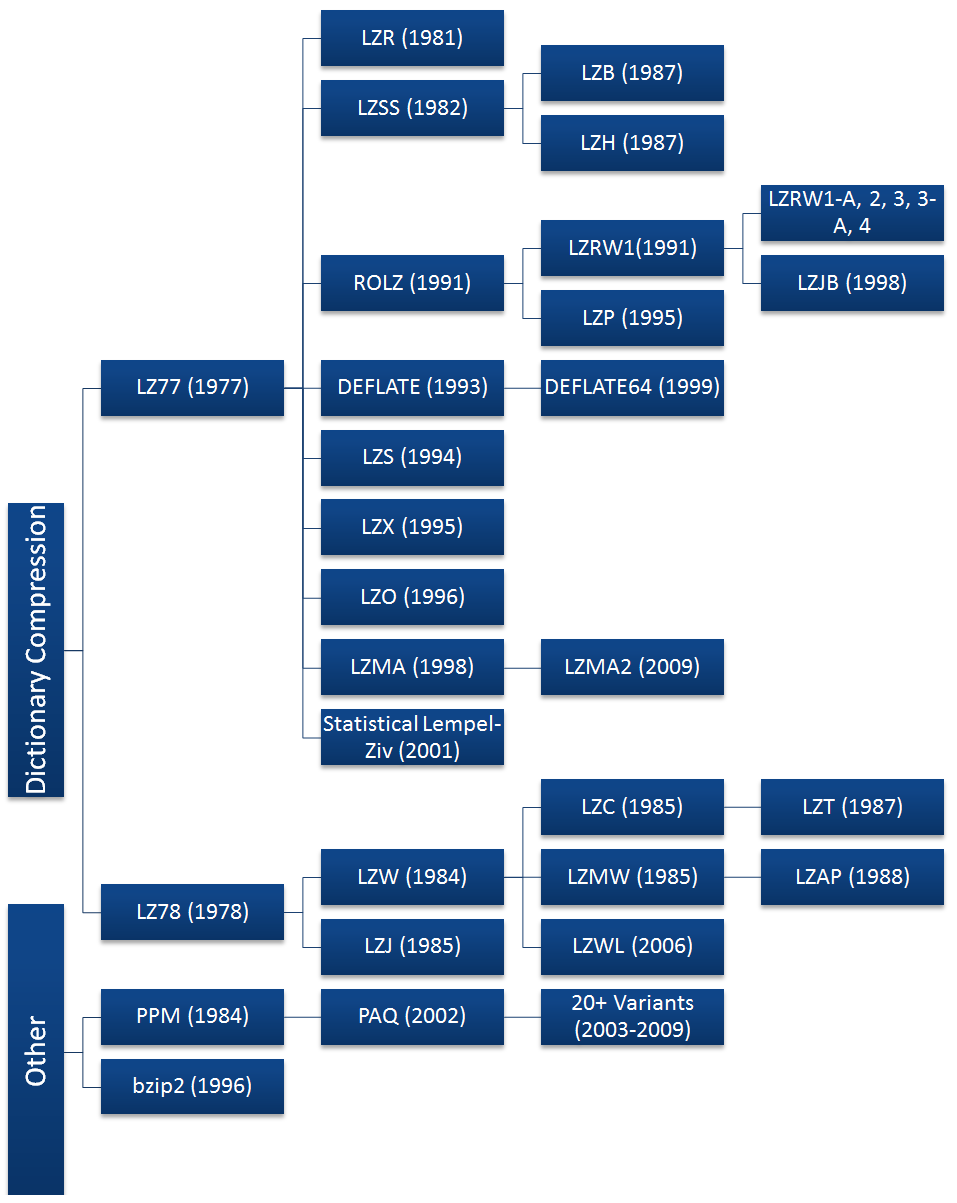


Рисунок 1.1 – Иерархия алгоритмов

**1.2 Программы для индивидуального пользования**

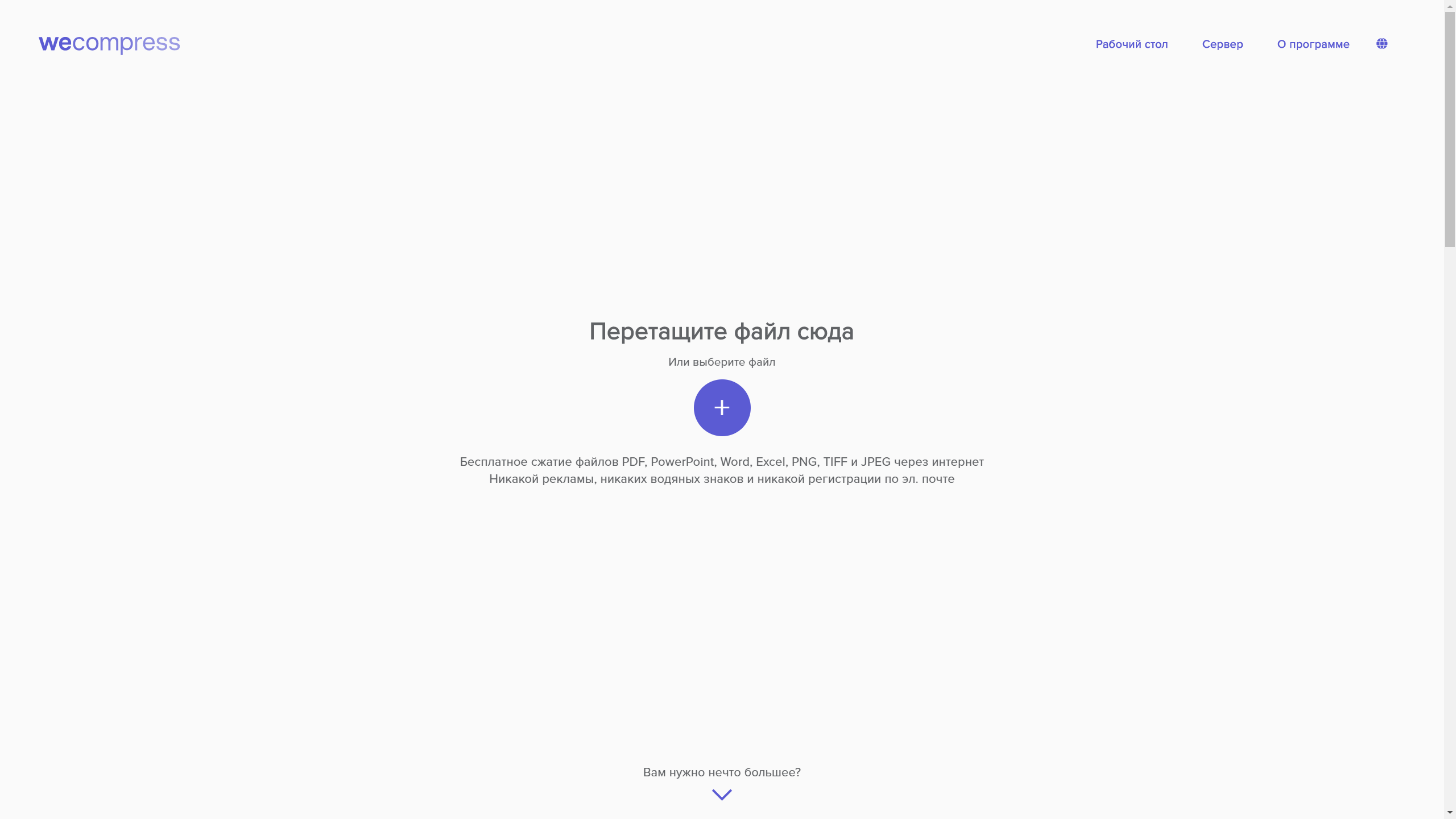
*Онлайн сервис «Wecompress»*[1].

Плюсы:

Отличный сайт, позволяющий быстро сжать PDF, PNG, TIFF и JPEG. Приятный и интуитивно понятный пользовательский интерфейс. Содержит вкладку с инструкцией по эксплуатации для пользователя. Переведен на все основные языки мира.

Минусы:

Онлайн программа, для доступа к ней необходимо подключение к сети, что не всегда является возможным. Сжатие происходит путем уменьшения качества файла без возможности восстановления.

Интерфейс изображен на рисунке 1.2 и 1.3.

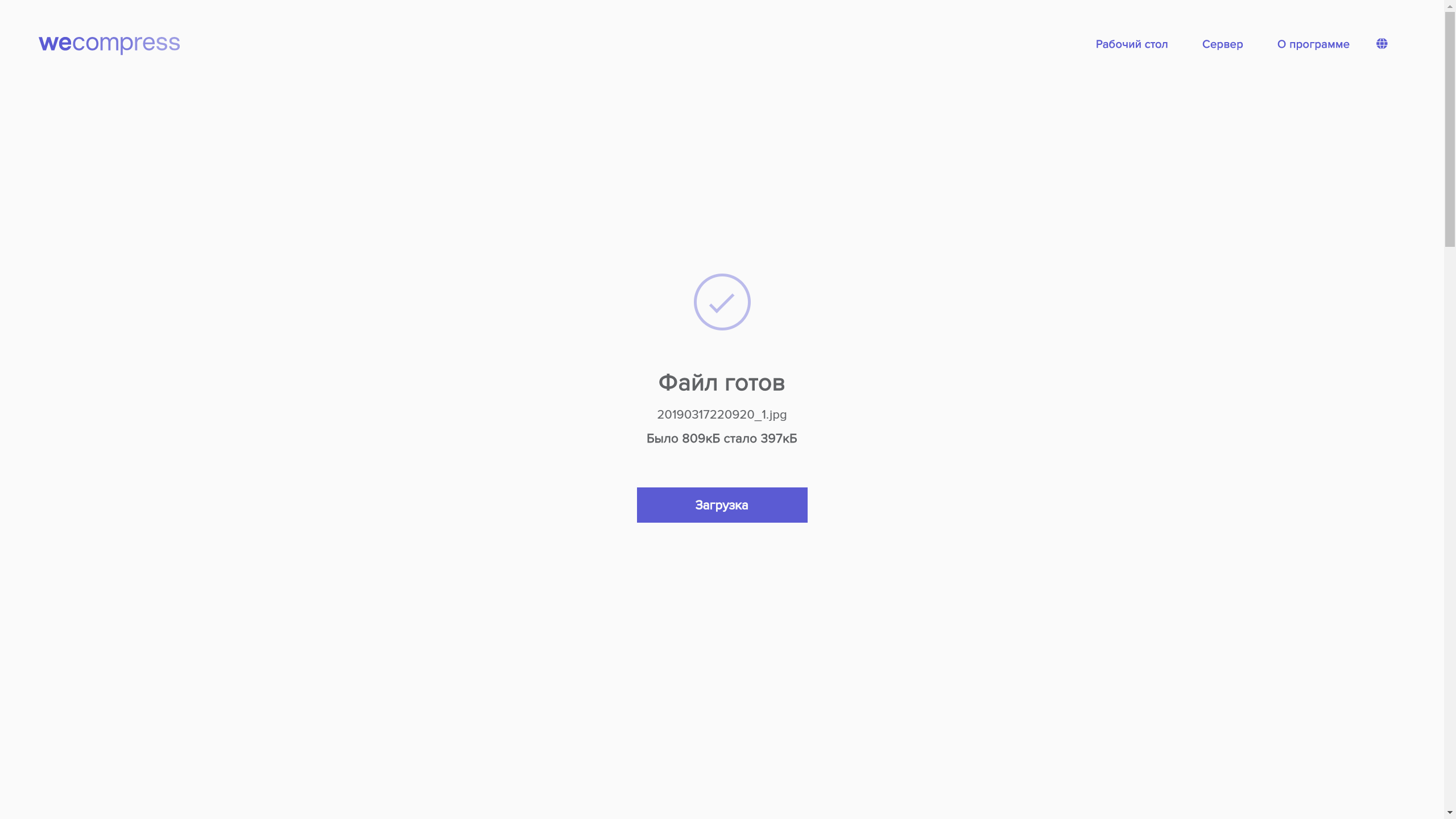
Рисунок 1.1 – Скриншот сайта “[www.wecompress.com](https://www.wecompress.com/ru/)” до сжатия файла

Рисунок 1.2 – Скриншот сайта “[www.wecompress.com](https://www.wecompress.com/ru/)” после сжатия файла

*Архиватор «WinRAR»*[2].

Плюсы:

* Разработчики поддерживают продукт и регулярно выпускают обновления;
* Создание архивов [RAR](https://ru.wikipedia.org/wiki/RAR) (с выбором формата — RAR4 или RAR5) и [ZIP](https://ru.wikipedia.org/wiki/ZIP) 2.0, их обновление и проверка целостности.
* Распаковка архивов [RAR](https://ru.wikipedia.org/wiki/RAR), а также [ARJ](https://ru.wikipedia.org/wiki/ARJ), [bz2](https://ru.wikipedia.org/wiki/Bzip2), [CAB](https://ru.wikipedia.org/wiki/CAB), [GZ](https://ru.wikipedia.org/wiki/GZ), [ISO](https://ru.wikipedia.org/wiki/ISO-%D0%BE%D0%B1%D1%80%D0%B0%D0%B7), [JAR](https://ru.wikipedia.org/wiki/JAR), [LZH](https://ru.wikipedia.org/wiki/LZH), [TAR](https://ru.wikipedia.org/wiki/TAR), [UUE](https://ru.wikipedia.org/wiki/UUE), [XZ](https://ru.wikipedia.org/wiki/XZ), [Z](https://ru.wikipedia.org/wiki/Z), [ZIP](https://ru.wikipedia.org/wiki/ZIP), [ZIPX](https://ru.wikipedia.org/wiki/WinZip), [7z](https://ru.wikipedia.org/wiki/7z), 001.
* Поддержка многоядерности на соответствующих [ЦП](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) при упаковке и распаковке.
* Полная поддержка имён файлов в [Юникоде](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4).
* Создание [самораспаковывающихся](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D1%80%D0%B0%D1%81%D0%BF%D0%B0%D0%BA%D0%BE%D0%B2%D1%8B%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B9%D1%81%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D0%B2) (SFX) архивов;
* Возможность управления из [командной строки](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0);
* Позволяет выбрать метод и степень сжатия.

Минусы:

Является платным по истечении пробного периода.

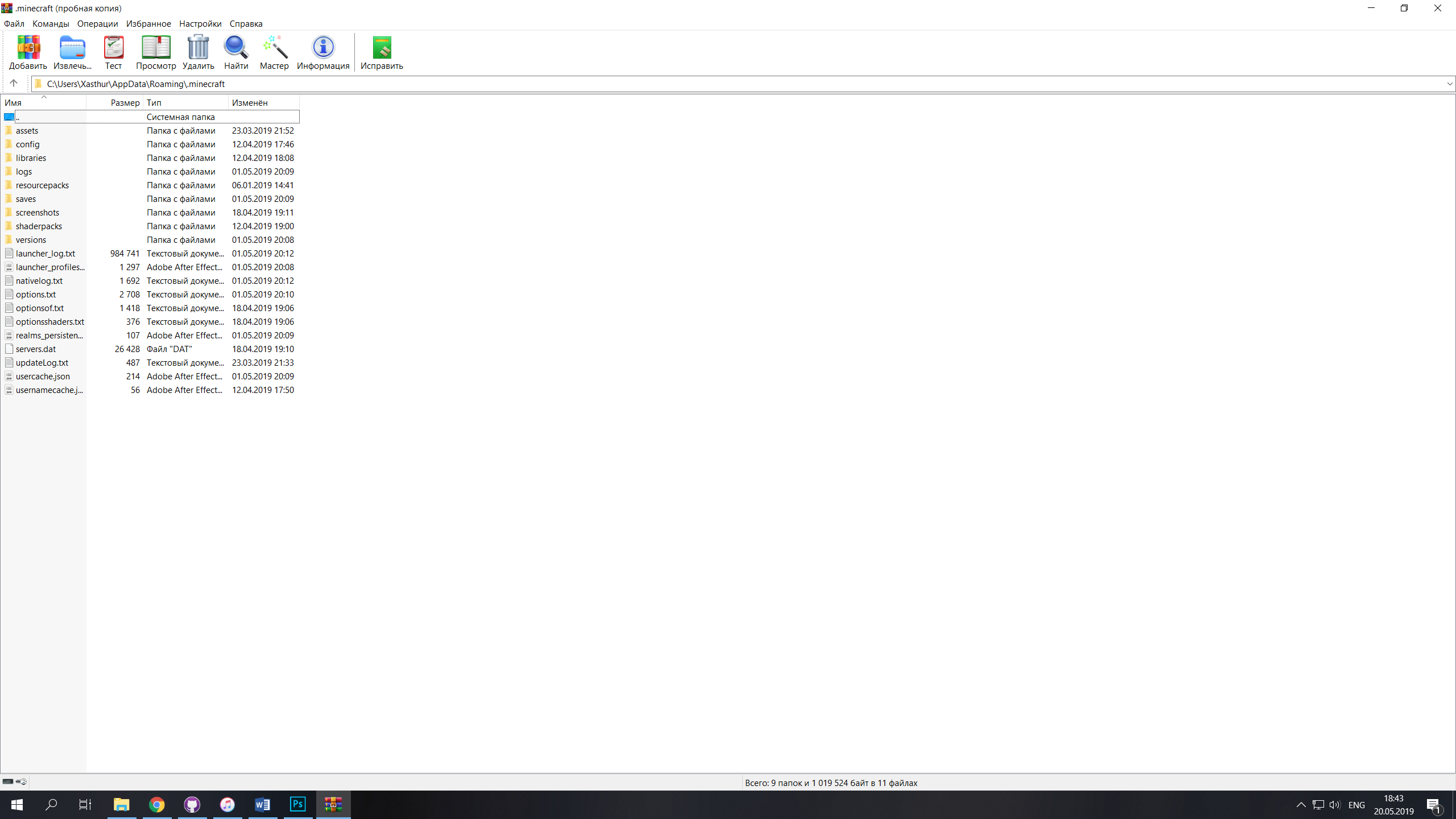
 Интерфейс программы показан на рисунках 1.3 и 1.4.

Рисунок 1.2 – Скриншот рабочего окна WinRAR

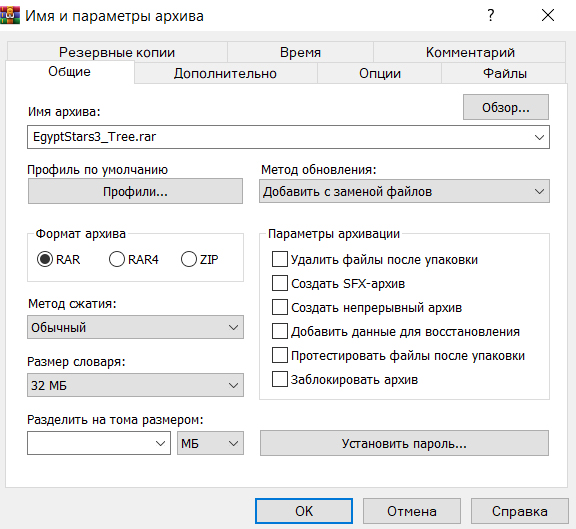


Рисунок 1.3 – Скриншот окна создания архива

**1.3 Описание популярных алгоритмов сжатия**

* *Алгоритм сжатия* [*RLE*](http://en.wikipedia.org/wiki/Run-length_encoding) *(Кодирование длин серий)*[3].

RLE— [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) [сжатия данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), заменяющий повторяющиеся символы (серии) на один символ и число его повторов. Серией называется последовательность, состоящая из нескольких одинаковых символов. При кодировании строка одинаковых символов, составляющих серию, заменяется строкой, содержащей сам повторяющийся символ и количество его повторов.

После применения алгоритма RLE строка ABCABCABCDDDFFFFFF превратится в 1A1B1C1A1B1C1A1B1C3D6F.

Допустим, реализация метода RLE для записи длин серий использует переменную символьного типа «Char». В такую переменную можно записать числа от 0 до 255 включительно. Но функция Chr(x) позволяет записать символ с кодом от 0 до 127 включительно. Как же быть, если длина серии равна 128 символам и более? В этом случае серию разделяют на части так, чтобы длина части не превышала 127 символов. Например, серия, состоящая из 256 символов «A», будет закодирована следующей строкой (256=127+127+2): 127A127A2A

Плюсы:

* Один из наиболее быстрых алгоритмов сжатия;
* Эффективен для сжатия текстовых файлов с длинными сериями повторяющихся символов;
* Эффективен для сжатия простых графических изображений, таких как [иконки](https://ru.wikipedia.org/wiki/%D0%97%D0%BD%D0%B0%D1%87%D0%BE%D0%BA_(%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0)) и графические рисунки;
* Звуковые данные, которые имеют длинные последовательные серии байт (такие как низкокачественные [звуковые семплы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%BF%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) могут быть сжаты с помощью RLE после того, как к ним будет применено [Дельта-кодирование](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D1%8C%D1%82%D0%B0-%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Минусы:

* Плохо подходит для изображений с плавным переходом тонов, таких как фотографии.
* Не подходит для сжатия обычных текстов, где повторений незначительное количество.
* *Алгоритм сжатия Хаффмана* [4].

Плюсы:

Минусы:

* *Алгоритм сжатия* [5].

Плюсы:

Минусы:

* *Алгоритм сжатия* [6].

Плюсы:

Минусы:

**1.4. Формирование требований к проектируемому программному средству**

Подробно изучив такие программы, как «Matrix Operations», «WolframAplpha», «MathCad», «Matrix Calculator Plus» я пришел к выводам, что должна из себя представлять программа, разработанная мною на языке Delphi в среде программирования «Borland Delphi 7» [5]:

1. математический функционал:

* программа должна позволять производить стандартные математические операции над матрицами, которые будет легко найти на панели инструментов;
* программа должна содержать некоторое количество теории по конкретному математическому действию, а также возможность просмотра алгоритма проведения данной операции;
* калькулятор, встроенный в приложение для удобства подсчетов;

1. прочие функциональные требования к программному средству:

* в случае возникновения ошибки ввода данных программа должна указать на ее место в программе;
* программа должна автоматически сохранять результаты вычислений во временный файл, автоматически создающийся в той же директории, где находится сама программа;
* приложение должно позволять открывать исходные данные из файла, распложенного в любой директории компьютера;
* программа должна позволять пользователю вручную сохранить результаты вычислений в любой временный или перманентный файл, находящийся в памяти компьютера или на съемном носителе;

1. требования для удобства эксплуатации:

* программа доступна без интернета (данное ПО для проведения операций над двумерными массивами чисел не будет требовать подключения к интернету, так как это не всегда возможно, сохранив данные программы на переносном носителе, программа всегда будет у вас под рукой);
* простой интуитивно понятный интерфейс;
* пользователь должен иметь возможность в любой момент времени обратиться к инструкции по эксплуатации приложения;
* краткие перманентные подсказки по использованию программы, выведенные на экран;
* интерактивные включения: звуковые оповещения при ошибках, примитивный аудиоплеер, тематические изображения.

**2 РАЗРАБОТКА АЛГОРИТМА**

**2.1** **Анализ требований к программному средству и разработка фф функциональных требований**

В результате анализа требований к программному средству были составлены следующие функциональные требования:

* корректность результата выполнения основных математических операций над матрицами, таких как:

1. сложение матрицы с числом и сложение матриц;
2. вычитание числа из матрицы, матрицы из числа и матрицы из матрицы;
3. умножение матрицы на число;
4. перемножение матриц;
5. возведение матрицы в целую положительную степень;
6. нахождение определителя матрицы;
7. транспонирование матрицы;

* изучение теории по необходимой математической операции и просмотра её алгоритма непосредственно в момент её выполнения, а также сокрытие данной информации в случае, если пользователь в ней не нуждается;
* минимально необходимая пользователю помощь в эксплуатации программного средства каждый раз при открытии приложения;
* вызов справки о программе и инструкции по ее эксплуатации из любой точки программы в любой необходимый момент времени;
* поддержка открытия пользовательских исходных данных из файлов форматов .rtf, .txt и ручного сохранения результатов вычислений в файл формата .txt;
* автоматическое сохранение результатов работы во временный файл в той же директории, в которой находится программа;
* быстрое очищения как какой-то определенной вкладки формы вычислений, так и всех ее полей;
* защита от потери данных при случайном закрытии формы вычислений, возможность их восстановления до момента полного закрытия приложения;
* быстрое переключение между вкладками формы вычислений при помощи сочетания горячих клавиш;
* быстрый переход на главную страницу формы вычислений из любой вкладки этой формы;
* проверка на корректность пользовательских данных, введенных как вручную, так и открытых из файла, и указание на ошибку в случае ее обнаружения;
* открытие программного калькулятора, выполняющего стандартные действия;
* сохранение истории вычислений, проведенных при помощи калькулятора, и операндов во временный файл и в поле истории на экране, путем нажатия горячих клавиш либо кнопки на калькуляторе;
* в случае необходимости - очищение истории вычислений калькулятора полностью или удаления конкретного элемента, при этом удаленный элемент будет исключен из временного файла;
* добавление элемента из временного файла в поле вычислений калькулятора;
* проверка запроса пользователя на проведение невозможных операций (например, деления на нуль);
* воспроизведение музыкального сопровождения и звуковых оповещений о той или иной ошибке (по желанию пользователя).

**2.2** **Разработка алгоритма программного средства**

На стадии проектирования приложения за основу концепции работы было принято выполнение выбранных пользователем действий.

Изначально вниманию пользователя представляется страница приветствия, где для продолжения работы программы ему необходимо нажать на кнопку «Начать работу». На этом этапе создаются временные файлы, в которые в последствие будут автоматически заноситься результаты вычислений, а также запоминается путь к исполняемому файлу (т.е. самому приложению).

После загружается окно, в котором нужно выбрать режим работы программы. В выбор режима работы входит определение того, желает ли пользователь слышать музыкальное сопровождение в ходе работы программы или нет. Первым будет вопрос о желании включения фоновой музыки (Да/Нет). В случае отказа пользователю будет задан вопрос о желании слышать звуковое сопровождение, если окажется, что он допустил ошибку при вводе данных вручную, из файла или вызывает недопустимую операцию или будет ясно, что данные введены верно и программа выполнила свое действие (Да/Нет). В зависимости от выбора на этом этапе будет строится дальнейший ход работы программы, однако это вовсе не повлияет на функционал и корректную работу программы.

Далее происходит загрузка формы, где пользователю необходимо выбрать: он хочет просмотреть инструкцию по эксплуатации программы или приступить к выполнению операций. После этого пользователь может выполнять некоторые действия на одной из форм, после закрытия которой он снова попадает на страницу выбора. Закрывая эту страницу, он автоматически закрывает и на этот момент недоступную и невидимую форму приветствия, тем самым закрывая и все приложения.

Укрупненная схема [6] программы представлена на рисунке 2.1.

Кроме того, программа не будет требовать установки на компьютер и подключения к Интернету. Таким образом вы сможете всегда носить ее с собой на любом съемном носителе, что несомненно является очень удобным.

Так же при разработке приложения решено использовать различные модули, перед каждым из которых поставлена определенная задача. Таким образом повышается уровень удобства для программиста в ориентировании по коду. Благодаря разделению на модули по задачам проще отлаживать, тестировать и обновлять приложение, что в реальных условиях является довольно важным фактором.



Рисунок 2.1 – Укрупненная схема работы программы

**2.3 Разработка алгоритма выполнения операций над матрицами**

При нажатии на клавишу операции на главной вкладке формы вычислений открывается соответствующая вкладка с несколькими обязательными и необязательными для заполнения компонентами, кнопками, а также теоретическими и алгоритмическими подсказками. Этими компонентами являются поля размерности матриц, непосредственно сами матрицы, число для проведения операции (если необходимо); кнопками – кнопка задания размерности матриц, кнопка выбора проведения операции с числом или матрицей (если необходимо), кнопка сокрытия алгоритмов и теории (не всегда), кнопка выполнения операции (рис. 6.6). После этого пользователю необходимо заполнить обязательные поля вкладки.

В случае внесения в поля некорректной информации, пользователю будет показано соответствующее сообщение с указанием на ошибку и, в зависимости от выбранного режима работы, прозвучит или нет определенное звуковое сообщение. Некоторые поля в случае некорректности введенных данных будут заполнены нулями, и операция будет проведена с такими данными. После проведения операции во временный файл, созданный в директории программы, будут занесены результаты вычислений, которые удаляются с выходом из приложения.

Для проведения операций над матрицами были использованы стандартные алгоритмы сложения, вычитания, умножения, транспонирования, возведения в степень матриц, а также нахождения определителя матрицы [7].

Алгоритм проведения операций над матрицами приведен на рисунке 2.2.



Рисунок 2.2 – Схема проведения операций над матрицами

Для защиты от потери данных в случае случайного закрытия страницы, заполненные поля не будут очищаться автоматически. Таким образом, это надо будет делать вручную.

**2.4 Разработка алгоритма открытия данных из файла и сохранения**

**результатов в файл вручную**

В целях экономии времени, а, следовательно, и повышения удобства использования приложения, был разработан алгоритм открытия и занесения в поля вкладок формы вычислений данных из пользовательских файлов. Открытие данных происходит только из файлов форматов .txt и .rtf. В ходе открытия так же происходит проверка данных на корректность, как и в случае ручного заполнения полей (звуковое сопровождение в случае выбора соответствующего режима работы так же воспроизводятся). После проведения операции результаты, как и в предыдущем случае записываются во временный файл. При открытии данных из файла поля вкладок очищаются автоматически.

Поскольку временные файлы существуют только во время выполнения программы, а часто необходимо хранить результаты вычислений длительное время, был разработан алгоритм ручного сохранения результатов в файл формата .txt. При этом результаты так же остаются сохраненными во временном файле.

Алгоритм открытия данных из файла и сохранения результата в файл вручную представлен на рисунке 2.3.



Рисунок 2.3 – Схема открытия данных из файла и ручного сохранения результата в выбранный файл

**2.5 Разработка алгоритма добавления и удаления результатов работы программного калькулятора**

При нажатии на главной странице формы вычислений на клавишу «Калькулятор», откроется отдельная форма с рабочим полем (рис. 6.7). На этом этапе в директории программы создается временный файл, в который будут занесены необходимые данные.

Для добавления содержимого поля ввода-вывода в историю вычислений, необходимо либо нажать на кнопку «А», либо сочетание горячих клавиш «alt + A».

При каждом новом нажатии индекс следующего элемента будет автоматически увеличен на единицу.

Для удаления элемента из истории и, следовательно, из файла необходимо кликнуть на этот элемент в поле истории. После этого элемент будет удален из списка истории, а счетчик индексов элементов уменьшиться на единицу. Удаление возможно производить до тех пор, пока счетчик индексов элементов не станет равным -1. Так же предусмотрена возможность быстрого очищения всего списка.

Алгоритм представлен на рисунке 2.4.



а) б)

Рисунок 2.4 – Схема добавления и удаления результатов работы программного калькулятора: а – добавление в историю; б – удаление из истории

**3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**3.1** **Разработка используемых данных**

1) Т.к. цель программы – проводить вычисления, связанные с числовыми матрицами, хранить результаты вычислений в памяти компьютера, как временно, так и постоянно, было решено использовать файлы форматов .txt и .rtf для хранения информации и загрузку данных в программу.

2) При выборе структуры данных, которая будет хранить в себе информацию о матрицах, результатах вычислений в виде матрицы, было учтено, что часто пользователь может неоднократно менять свои решения по их размерности, заполнению и может подвергать матрицы различным изменениям в ходе работы программы, что некоторые математические операции так же требуют изменения размерности матриц.

В связи с вышесказанным было проведено сравнение эффективности использования статического и динамического массивов. Во время анализа было установлено, что в случае статического массива в отличие от динамического память выделяется на этапе трансляции и под максимально возможное для задачи количество элементов массива, тогда как в динамическом случае на этом этапе память выделяется под указатели на элементы. Таким образом, статический массив не решает вопроса с эффективным изменением размеров операндов, поскольку нам придется выделять под программу память, которая может быть невостребованной в ходе работы приложения и замелить работу программы, что крайне нежелательно.

Также во время разработки структур данных был рассмотрен вариант использования динамического двунаправленного или однонаправленного списка путем их сравнения с динамическим массивом. Анализ [8] показал, что:

* несмотря на то, что списки – это очень гибкая структура, позволяющая проводить поиск, менять свой размер, удалять и проводить вставку элемента в любой позиции, в данном приложении это будет не так востребовано, поскольку в ходе работы программы не придется удалять, искать или вставлять элементы в середину списка, будет лишь проводиться вставка нового элемента в его конец, чтение всех данных и их вывод на экран, что гораздо проще и удобней реализуется при помощи динамического массива;
* при работе со списком не понадобится непрерывная область памяти, поэтому возникает необходимость явной работы с указателями, следовательно, придется «платить» за удобство использования списка выделением дополнительной памяти под ссылки, что также является нерентабельным в этой программе, поскольку преимущества списка использованы не будут.

Таким образом, на основе проведенных исследований было установлено, что наиболее благоприятным для решения поставленной задачи будет использование такой структуры данных, как динамический массив типа «real», потому что:

* в случае использования статического массива нам придется выделять под программу память, которая может быть невостребованной в ходе работы приложения и замелить работу программы, в то время как динамическая структура даст нам возможность резервировать строго столько памяти, сколько необходимо;
* несмотря на все преимущества динамической цепочки, в данной задаче использование массива будет более удобным, поскольку эти преимущества задействованы при решении не будут.

Структура динамического массива и однонаправленного списка представлены на рисунках 3.1, 3.2 и 3.3 соответственно.

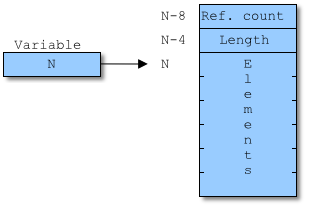


Рисунок 3.1 – Структура динамического массива

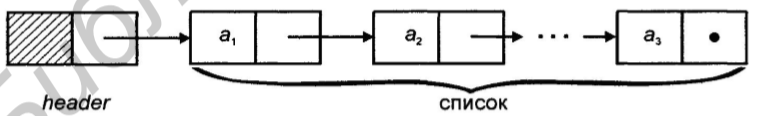


Рисунок 3.2 – Структура однонаправленного списка

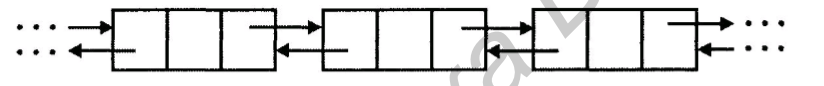
3)Поскольку в приложении реализован пример простейшего аудиоплеера, было решено, что для хранения списка плейлиста будет использован двунаправленный закольцованный список (рис. 3.3) типа запись TPlayList (рис. 3.4).

Рисунок 3.3 – Структура двунаправленного списка

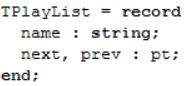


Рисунок 3.4 – Объявление типа TPlayList

**3.2** **Разработка схемы работы системы**

В приложении помимо методов объектов и компонентов были реализованы собственные подпрограммы разработчика. Перечень собственных подпрограмм представлен в таблице 3.1

Таблица 3.1 – Собственные подпрограммы разработчика

|  |  |
| --- | --- |
| Название подпрограммы | Назначение |
| **Модуль uActions** | |
| SaveTable (процедура) | Сохранение данных таблицы в файл |
| SaveEdit (процедура) | Сохранение данных поля в файл |
| CheckString (процедура) | Проверка элемента таблицы на корректность |
| CheckEdit (процедура) | Проверка данных поля на корректность |
| SetEdit (процедура) | Проверка и преобразование данных поля в нужный тип |
| Clean (процедура) | Очистка таблицы и полей вкладки |
| Add (процедура) | Сложение матриц (матрицы и числа) |
| Sub (процедура) | Вычитание матрицы (матрицы и числа) |
| Determinate (процедура) | Нахождение определителя |
| Det (функция): real | Нахождение определителя |
| MulMatrix (процедура) | Перемножение матрицы |
| MultNumb (процедура) | Умножение матрицы на число |
| Pow (процедура) | Возведение матрицы в степень |
| Transpose (процедура) | Транспонирование матрицы |
| Music (процедура) | Воспроизведение звуковых предупреждений |
| **Модуль uCalculator** | |
| Doing (процедура) | Выполнение операции калькулятора |
| DoOperation (процедура) | Выполнение операции калькулятора |

Таблица 3.1 (продолжение)

|  |  |
| --- | --- |
| Название подпрограммы | Назначение |
| NumbShow (процедура) | Обработка строчного символа и приведение в число |
| **Модуль uOperations** | |
| CleanAll | Очистка всех полей всех вкладок формы вычислений |
| Click1 | Открытие вкладки операции |
| Enabling | Запрет на использование кнопки «Выполнить» |
| ExitOpen | Остановка открытия данных из файла при некорректности ввода |
| MenuFile | Активация кнопок пункта «Файл» программного меню |
| ShowAndExit | Невидимость вкладок формы операций |
| **Модуль uSound** | |
| CreateList | Создание плейлиста на основе закольцованного списка |

**4 ОБОСНОВАНИЕ ТЕХНИЧЕСКИХ ПРИЕМОВ**

**ПРОГРАММИРОВАНИЯ**

Программа включает в себя 11 различных модулей:

* UAboutProgramm

Эта форма отвечает за показ информации о программе, ее разработчике, его научном руководителе, контактах разработчика. На ней находится компонент показа текста (TMemo) в режиме Read-only, фотография разработчика (TImage), метка с названием приложения (TLabel). Вызывается со страницы UOperations;

* UActions

Модуль, отвечающий за основную логику приложения и содержащий в себе такие функции, как функция сложения матрицы с матрицей и матрицы с числом, вычитания числа из матрицы, матрицы из числа и матрицы из матрицы, умножение матрицы на число, перемножение матриц, возведение матрицы в положительную целую степень, нахождение определителя матрицы, транспонирование матрицы; также отвечает за проверку корректности введенных данных, очистку полей главной формы и сохранение результатов вычислений в файлы в памяти компьютера или на съемный носитель. Вызывается из UOperations;

* UCalculator

Форма, реализующая работу простейшего калькулятора, с тем отличием, что результат вычислений можно сразу использовать для добавления в список (историю) вычислений. Вызывается со страницы UOperations. Реализуется с помощью компонентов типов TButton, TEdit, TListBox;

* UChoise

Форма, отвечающая за выбор направления продолжения работы: перейти к вычислениям или же прочитать инструкцию по эксплуатации. По сути является первой формой, где пользователю необходимо сделать выбор в направлении дальнейшего использования приложения. Реализуется при помощи компонентов типов TButton и TImage. Вызывается со страницы USound;

* UHi

Страница приветствия, появляющаяся первой после запуска приложения. Также служит для выхода из приложения, поскольку ее закрытие служит сигналом для сворачивания программы. Во время работы приложения находится в пассивном режиме, однако при этом остается видимой, доступной и возможной для использования. Скрывается другими открытыми формами. Реализована при помощи компонента TImage и единственной кнопки (TButton);

* UInstruction

Форма текстовой инструкции по эксплуатации. Содержит единственный

компонент типа TRichEdit, в который загружается информация из текстового файла с инструкцией, созданного заранее в директории программы. Вызывается со страницы UChoise и UOperations;

* UMultMatr

Форма, cсодержащая в себе алгоритм выполнения перемножения матриц. Реализована только изображением типа TImage. Вызывается со страницы UOperations;

* UOpenUChoise

Форма, представляющая из себя одну кнопку (TButton), использующуюся для полной очистки полей страницы UOperations. Форма открывается при вызове формы UOperations со страницы UChoise. Остается активной и готовой к использованию пока открыта форма UOperations, при ее закрытии также автоматически закрывается;

* UOperations

Главная форма программы, содержащая в себе путь ко всем математическим операциям, программному калькулятору, дочерним формам. Форма реализована при помощи компонента типа TPageControl, содержащего 8 вкладок (TTabSheet), на каждой из которых в зависимости от направления ее использования находятся компоненты типов TMemo в режиме Read-only, TButton, TStringGrid (частично в режиме Read-only), TEdit (частично в режиме Read-only), TImage, TMainMenu, TLabel, TMediaPlayer. Также из любой вкладки (TTabSheet) через меню (TMainMenu) возможен доступ к диалоговым окнам (TOpenDialog, TSaveDialog). Задействует все оставшиеся формы. Вызывается из UChoise.

* UPlayerOff

Форма, представляющая из себя одну кнопку (TButton), предназначающуюся

для отключения аудиоплеера формы USound. Вызывается при включении плеера из формы USound и является активной, пока плеер включен, при его выключении скрывается;

* USound

Форма выбора режима работы (беззвучного/со звуком/частичный звук). Вызывается из формы UHi. После вызова, в случае выбора пользователя, создается плейлист аудиоплеера этой формы, а также подаются команды на аудиоплеер формы UOperations. Реализуется при помощи компонентов типов TButton, TLabel, TMediaPlayer.

**5 ТЕСТИРОВАНИЕ**

**5.1 Тестирование математической составляющей функционала ПО**

Результаты тестирования математической составляющей функционала программного средства представлены в таблице 5.1. Подтверждение корректности проводимого тестирования представлено на рисунках [5.1]-[5.15].

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| [1] | Сложение матриц  1.Ввод корректных значений  2.Нажатие на кнопку «Выполнить» | Корректный результат [5.1] | Корректный результат [5.2] |
| [2] | Вычитание матрицы из числа  1.Ввод корректных значений  2.Нажатие на кнопку «Выполнить | Корректный результат [5.3] | Корректный результат [5.4] |
| [3] | Умножение матрицы на число  1.Ввод корректных значений  2.Нажатие на кнопку «Выполнить | Корректный результат [5.5] | Корректный результат [5.6] |
| [4] | Перемножение матриц  1.Ввод корректных значений  2.Нажатие на кнопку «Выполнить | Корректный результат [5.7] | Корректный результат [5.8] |
| [5] | Возведение матрицы в степень  1.Ввод корректных значений  2.Нажатие на кнопку «Выполнить | Корректный результат [5.9] | Корректный результат [5.10] |
| [6] | Нахождение определителя матрицы  1.Ввод корректных значений  2.Нажатие на кнопку «Выполнить | Корректный результат [5.12], [5.11] | Корректный результат [5.13] |

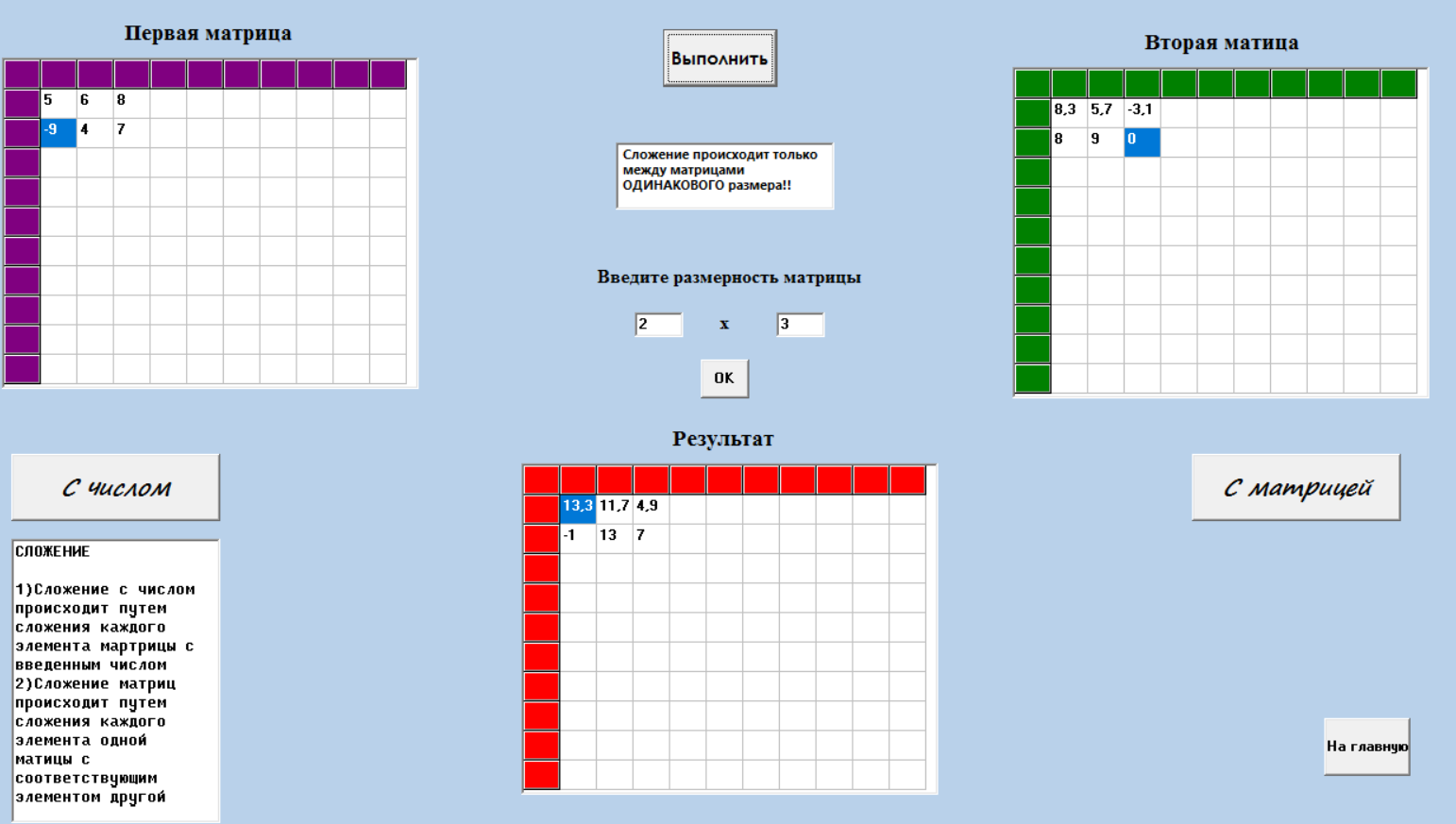
Таблица 5.1 – Тестирование математического функционала ПО

Таблица 5.1 (продолжение)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| [7] | Транспонирование матрицы  1.Ввод корректных значений  2.Нажатие на кнопку «Выполнить» | Корректный результат [5.14] | Корректный результат [5.15] |

Подтверждение корректности проводимого тестирования:

1. Тест [1] – сложение матриц:



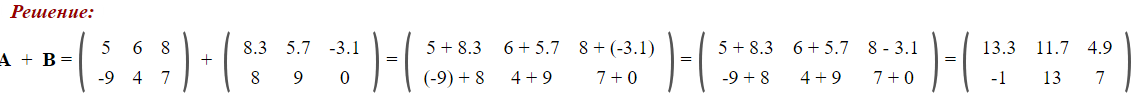
Рисунок [5.2] – Результат проведения операции сложения

Рисунок [5.1] – Результат проверки проведения операции сложения

1. Тест [2] – вычитание матрицы из числа:

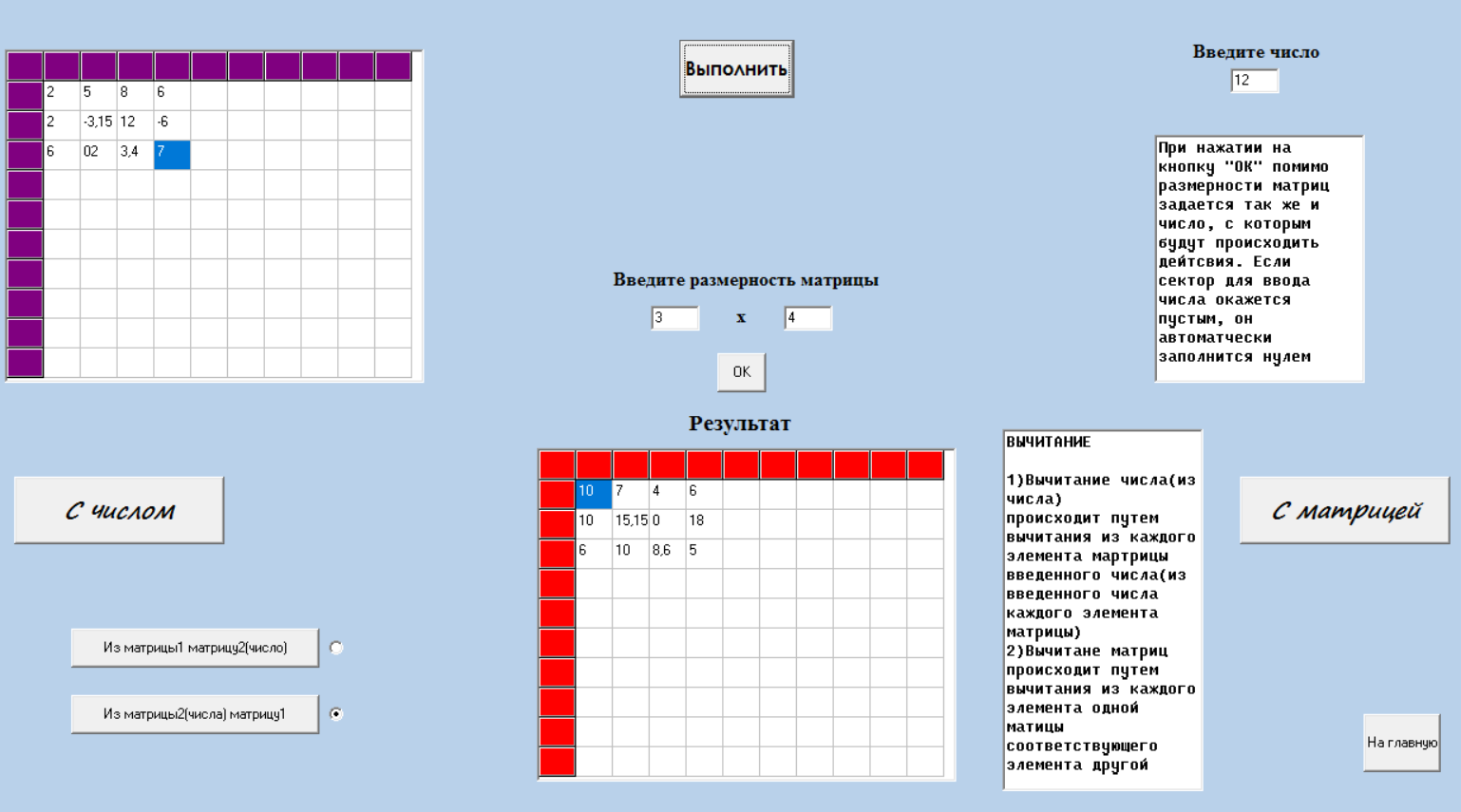


Рисунок [5.4] – Результат проведения операции вычитания матрицы из числа

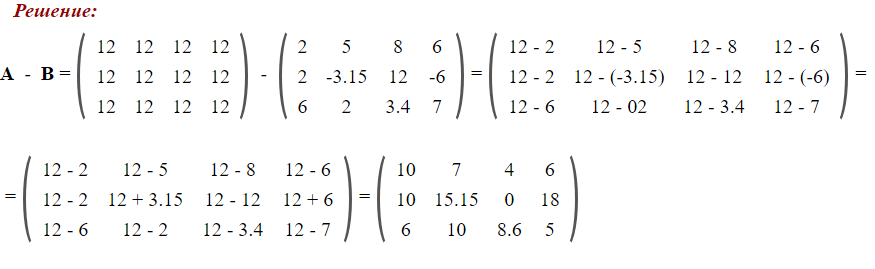


Рисунок [5.3] – Результат проверки проведения операции вычитания матрицы из числа

1. Тест [3] – умножение матрицы на число:

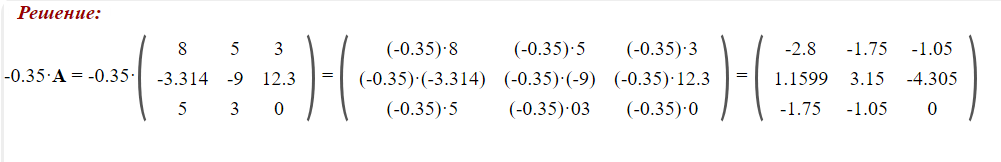


Рисунок [5.5] – Результат проверки проведения операции умножения матрицы на число

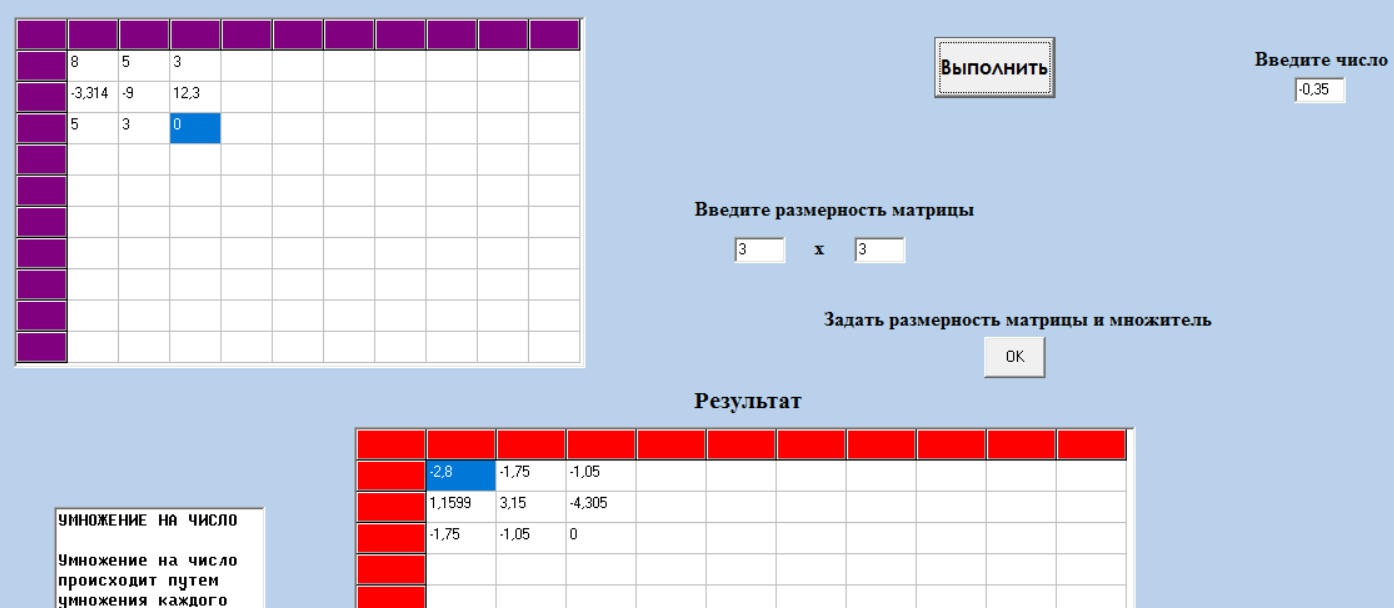


Рисунок [5.6] – Результат проведения операции умножения матрицы на число

1. Тест [4] – перемножение матриц:

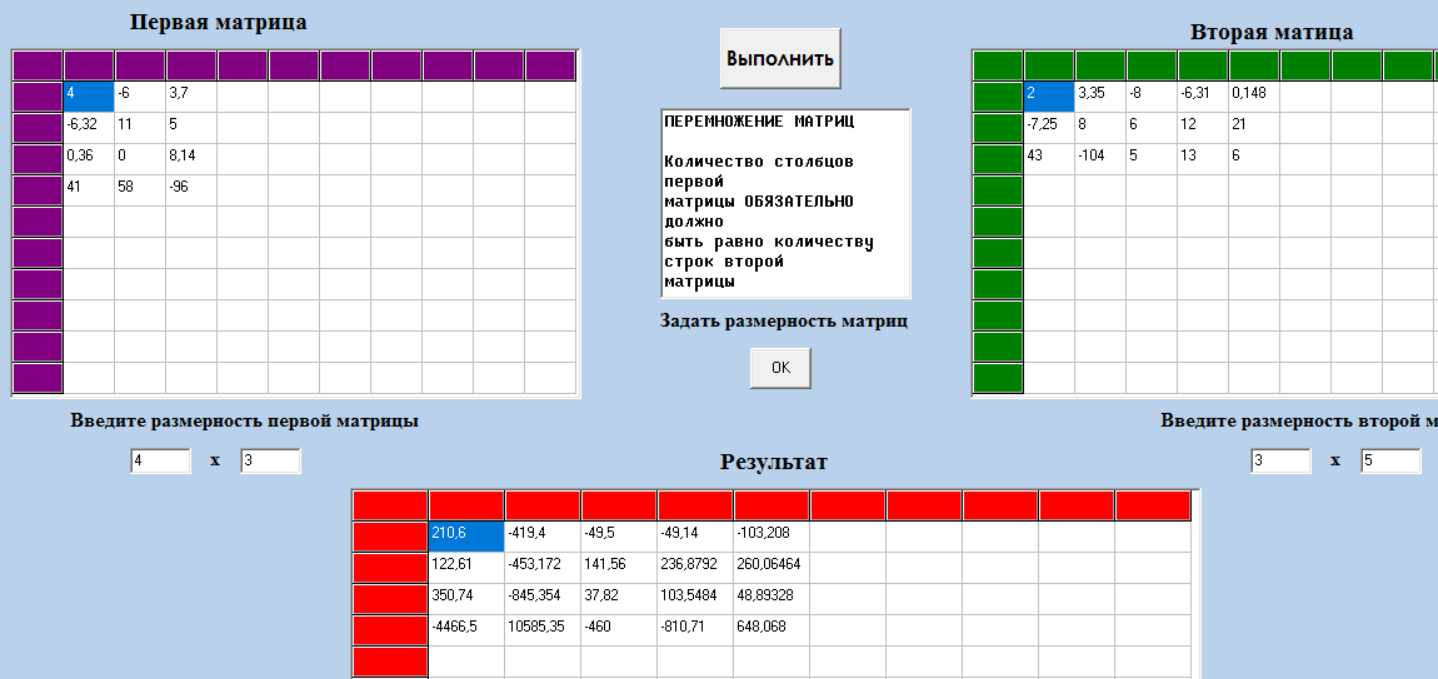


Рисунок [5.8] – Результат работы операции перемножения матриц

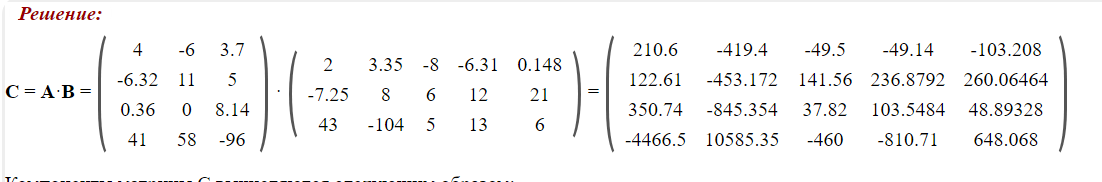


Рисунок [5.7] – Результат проверки операции перемножения матриц

1. Тест [5] – возведение матрицы в степень:

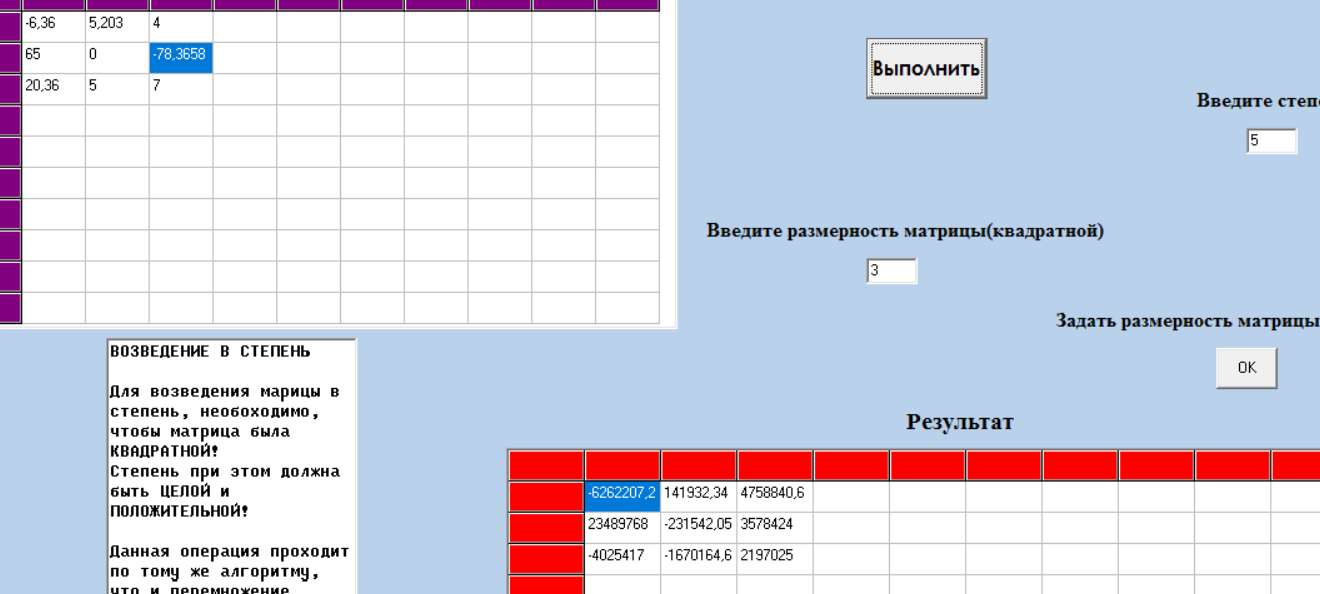


Рисунок [5.10] – Результат работы операции возведения матрицы в степень

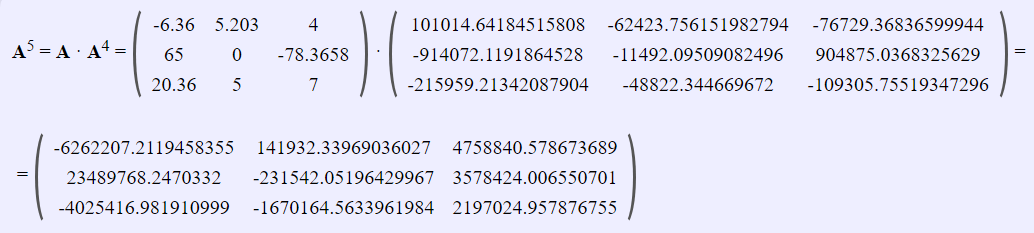


Рисунок [5.9] – Результат проверки операции возведения матрицы в степень

1. Тест [6] – нахождение определителя матрицы:

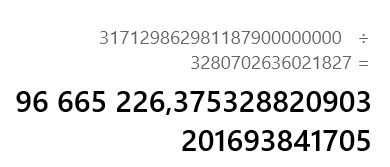


Рисунок [5.12] – Результат проверки операции

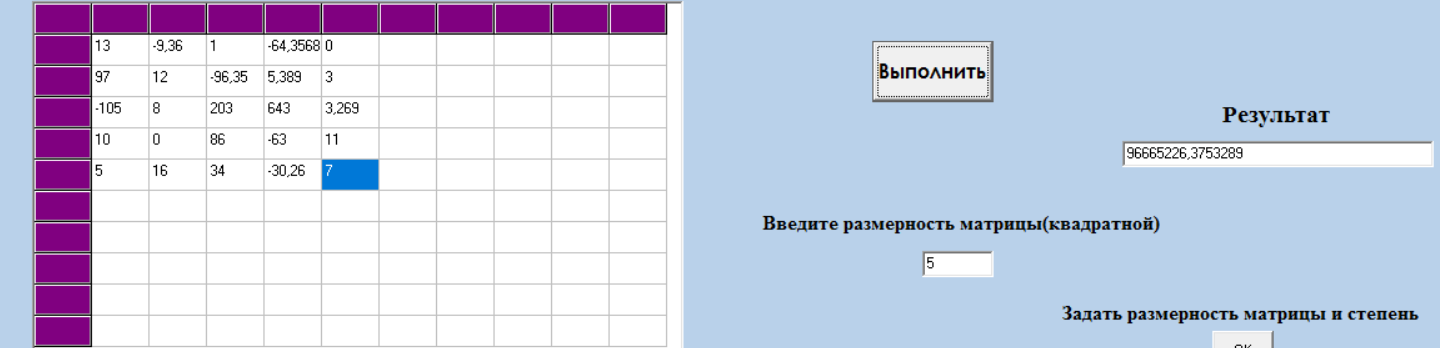
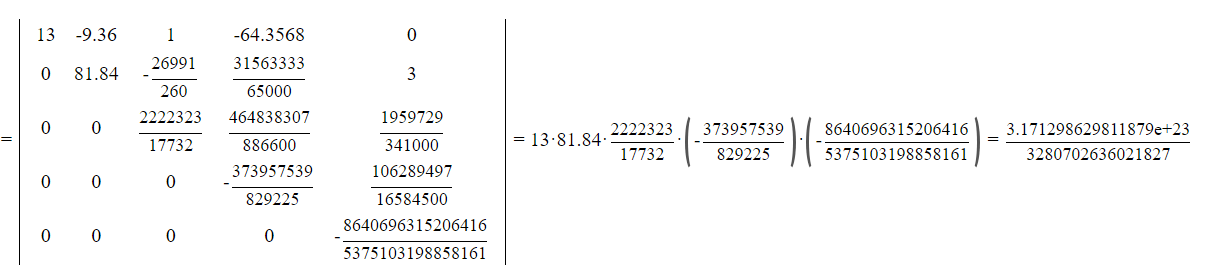
Рисунок [5.13] – Результат работы операции нахождения определителя матрицы

Рисунок [5.11] – Результат проверки операции нахождения определителя матрицы

1. Тест [7] – транспонирование матрицы:

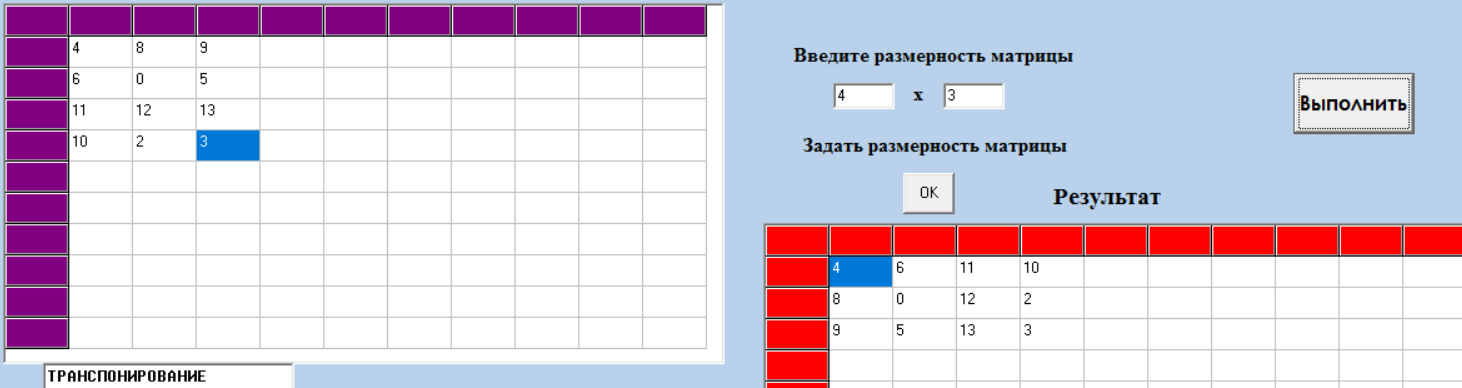


Рисунок [5.15] – Результат операции транспонирования матрицы



Рисунок [5.14] – Результат проверки операции транспонирования матрицы

Таким образом, программное средство успешно прошло тестирование математического функционала, и, на основе полученных результатов, можно сделать вывод, что приложение корректно выполняет часть основную поставленной перед ним задачи.

**5.2 Тестирование прочих функций программы**

Результаты тестирования прочих функций представлены в таблице 5.2.

Таблица 5.2 – Тестирование прочих функций программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| [1] | Открытие данных из файла  1.Нажатие кнопки «Открыть» пункта «Файл» программного меню  2.Появление диалогового окна  3.Выбор файла, открытие данных | Ввод данных в пустые поля в установленном порядке | Ввод данных в пустые поля в установленном порядке |
| 2 | Автоматическое сохранение результата в файл  1.Нажатие кнопки «Выполнить»  2.Вычисление результата  3.Автосохранение | Появление новой записи с результатом во временном файле | Появление новой записи с результатом во временном файле |
| 3 | Сохранение результата вручную  1.Нажатие кнопки «Сохранить» пункта «Файл» программного меню  2.Появление диалогового окна  3.Выбор файла | Появление новой записи с результатом в выбранном файле | Появление новой записи с результатом в выбранном файле |

Таблица 5.2 (продолжение)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| 4 | Сохранение записи в историю вычислений калькулятора  1.Нажатие на кнопку «А»  2.Добавление значения из поля | Появление во временном файле и поле формы новой записи с указанными данными | Появление во временном файле новой записи с указанными данными |
| 5 | Удаление записи из истории вычислений калькулятора  1.Установка выбора на «удаление»  2.Клик на нужный элемент | Удаление из временного файла и поля формы указанной записи | Удаление из временного файла и поля формы указанной записи |
| 6 | Добавление записи из истории калькулятора в поле вычислений  1.Установка выбора на «добавление»  2.Клик на нужный элемент | Появление элемента в поле вычислений | Появление элемента в поле вычислений |
| 7 | Заполнение полей некорректными данными  1.Ввод некорректных данных  2.Нажатие кнопки обработки | Просьба проверки введенных данных, звуковое сообщение (в определенном режиме) | Просьба проверки введенных данных, звуковое сообщение (в определенном режиме) |
| 8 | Открытие некорректных данных из файла  1.Смотрите тест [1] | Просьба проверки введенных данных, звуковое сообщение (в определенном режиме) | Просьба проверки введенных данных, звуковое сообщение (в определенном режиме) |
| 9 | Ввод отрицательных чисел в поля размерности и степени  1.Ввод данных  2.Нажатие кнопки обработки | Замена введенных чисел нулями | Замена введенных чисел нулями |

Таблица 5.2 (продолжение)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| 9 | Ввод числа с плавающей точкой в поля размерности и степени  1.Ввод числа в поле  2.Нажатие кнопки обработки | Обрезание цифр после запятой с преобразованием числа в целое | Обрезание цифр после запятой с преобразованием числа в целое |
| 10 | Ввод разных значений сопряженных строк и столбцов при перемножении матриц  1.Ввод чисел в поле  2.Нажатие кнопки обработки | Замена второго числа на первое | Замена второго числа на первое |
| 11 | Отсутствие данных в полях  1.Нажатие кнопки обработки | Заполнение полей нулями | Заполнение полей нулями |
| 12 | Некорректный ввод в программный калькулятор  1.Попытка ввода не числовых значений | Невозможно | Невозможно |
| 13 | Деление на ноль  1.Ввод в поле любого числа  2.Нажатие на знак деления  3.Ввод в поле нуля  4.Нажатие на знак «равно» | Сообщение о невозможности проведения операции, звуковое сообщение (зависит от режима) | Сообщение о невозможности проведения операции, звуковое сообщение (зависит от режима) |
| 14 | Автоматическая очистка полей всех вкладок  1.Переход на форму очищения полей всех вкладок  2.Нажатие на кнопку «Очистить всё» | Поля очищены | Поля очищены |

Таблица 5.2 (продолжение)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| 15 | Автоматическая очистка полей конкретной вкладки  1.Нажатие кнопки «Открыть» пункта «Файл» программного меню | Поля очищены | Поля очищены |
| 16 | Корректирование полей теории и алгоритмов  1.Попытка установить курсор на поле | Невозможно | Невозможно |
| 17 | Вычисления на программном калькуляторе  1.Нажатие на клавиши цифр  2.Нажатие на клавиши операций  3.Нажатие на клавиши цифр  4.Нажатие на «равно»  5.Вывод результата в поле | Полученные  результаты корректны | Полученные результаты корректны |
| 18 | Редактирование полей результата  1.Попытка установить курсор на поле | Невозможно | Невозможно |
| 19 | Выполнение операции через программное меню  1.Нажатие на клавиши программного меню  2.Ожидание результата в зависимости от клавиши | Операция выполнена | Операция выполнена |
| 20 | Открытие инструкции пользователя из файла  1.Открытие формы инструкции  2.Загрузка инструкции  3.Отображение инструкции | Инструкция открыта | Инструкция открыта |

Таблица 5.2 (продолжение)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| 21 | Открытие алгоритма  1.Нажатие на кнопку «Алгоритм» | Алгоритм доступен и открыт | Алгоритм доступен и открыт |
| 22 | Скрытие алгоритма  1.Нажатие на кнопку «Скрыть» | Алгоритм скрыт | Алгоритм скрыт |
| 23 | Включение плеера  1.Нажатие на кнопку «Да» на форме выбора режима на вопросе о включении плеера | Аудиоплеер включен | Аудиоплеер включен |
| 24 | Выключение плеера  1.Переход на форму выключения плеера  2.Нажатие на кнопку выключения | Аудиоплеер выключен | Аудиоплеер выключен |
| 25 | Включение аудио оповещений  1.Нажатие на кнопку «Да» на форме выбора режима на вопросе о включении оповещений  2.Совершение действия с матрицами | Оповещение воспроизвелось | Оповещение воспроизвелось |
| 26 | Невключение аудио оповещений  1.Нажатие на кнопку «Нет» на форме выбора режима на вопросе о включении оповещений  2.Совершение действия с матрицами | Оповещение не воспроизвелось | Оповещение не воспроизвелось |

Таким образом, программное средство успешно прошло все этапы тестирования и полностью готово к эксплуатации.

**6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Для того чтобы начать работу с приложением необходимо запустить MatrixCalculator.exe, а также обязательно наличие файла «Инструкция пользователя» в директории приложения.

После запуска первой откроется страница приветствия (рис. 6.1).

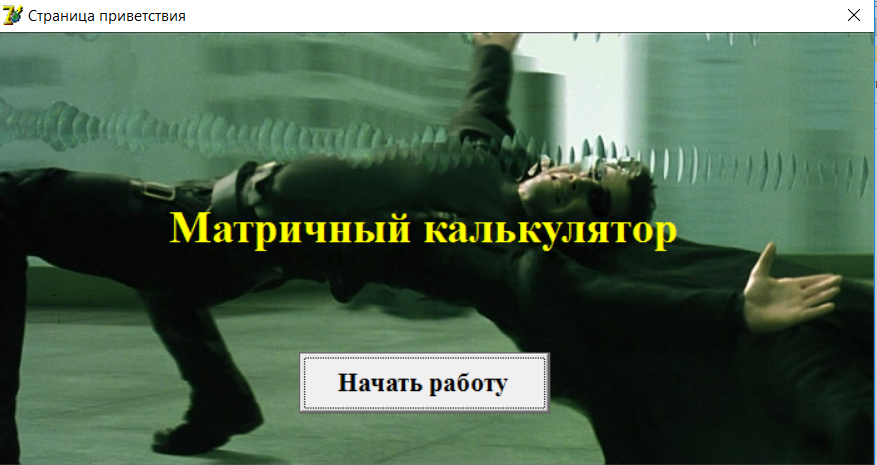


Рисунок 6.1 − Страница приветствия

Для продолжения работы приложения пользователю необходимо нажать на кнопку «Начать работу». После этого вниманию пользователя предстанет форма выбора аудио режима работы (рис. 6.2а). В зависимости от выбора откроется или форма выбора (рис. 6.3) или другие компоненты формы выбора аудио режима (рис. 6.2б). Если пользователь нажмет «Да», то включится аудиоплеер со втроенным плейлистом. Так же откроется форма выключения плеера (рис. 6.10)

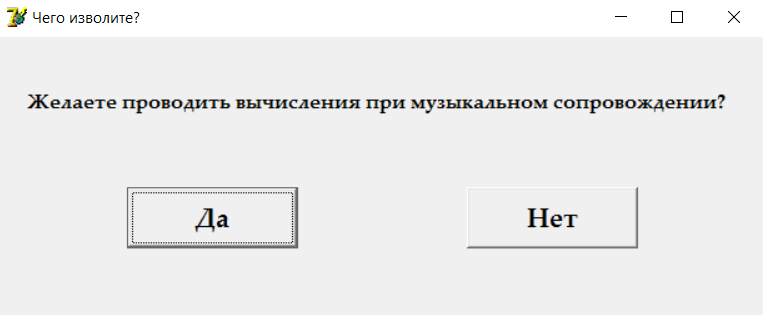


Рисунок 6.2а – Форма выбора аудио режима

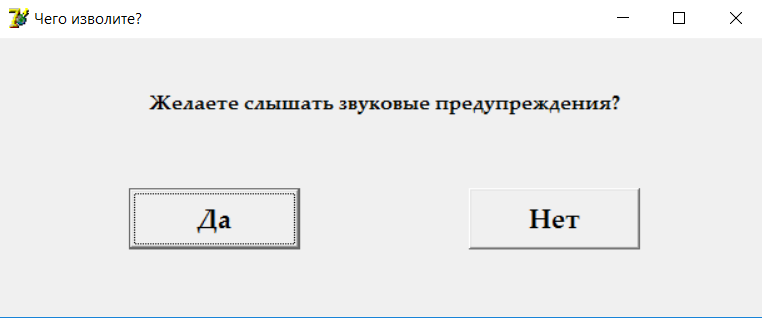


Рисунок 6.2б – Другие компоненты формы выбора аудио режима

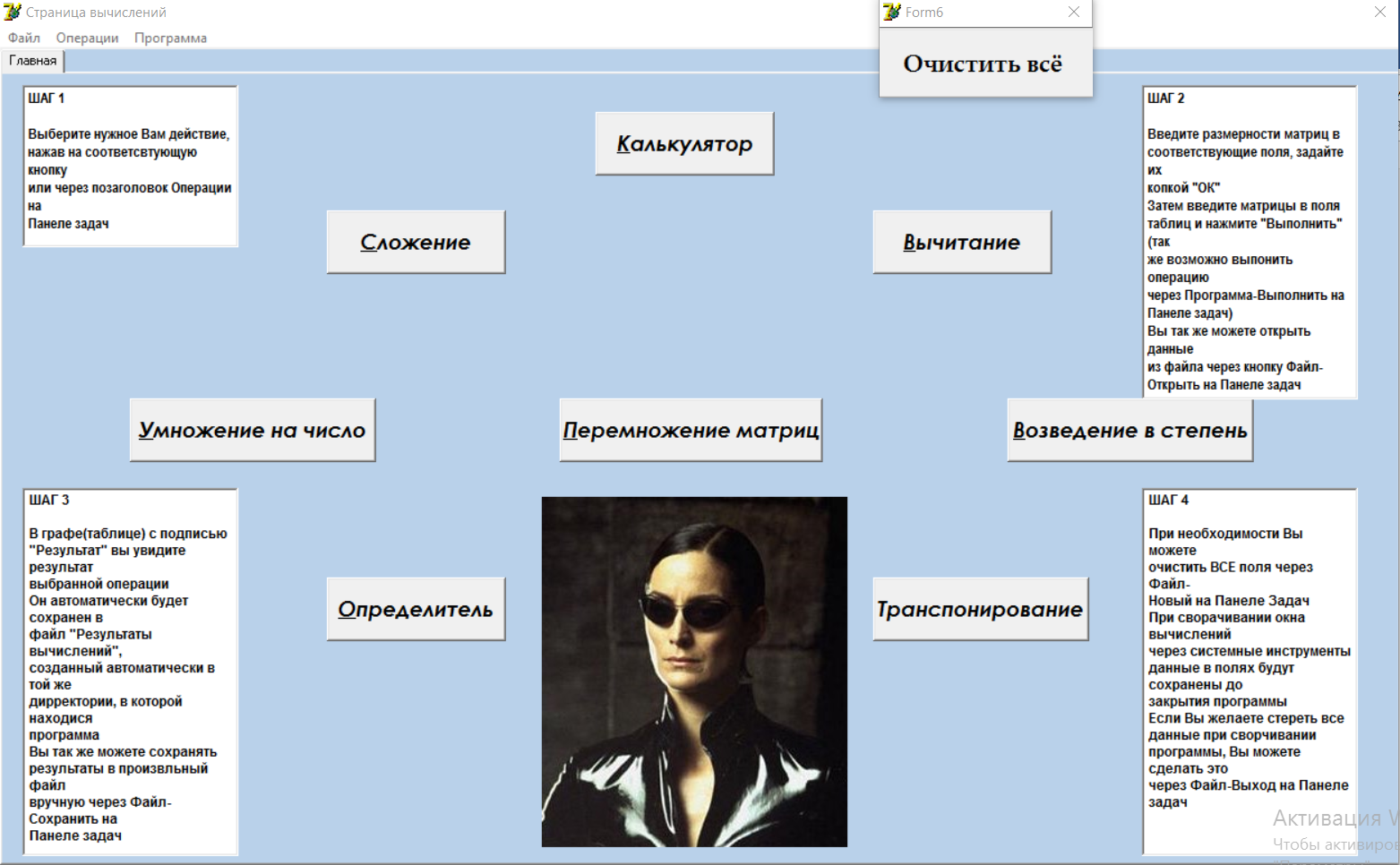
В случае нажатия на «Да» при совершении операций или вводе некорректных данных пользователь услышит звуковое сообщение об этом.

После нажатия любой из кнопок формы, показанной на рисунке 6.2б, откроется форма выбора (рис. 6.3).



Рисунок 6.3 − Страница выбора

При нажатии на кнопку «Справка» пользователь перейдет на открывшуюся форму, содержащую в себе текстовую инструкцию по эксплуатации приложения. Если же пользователь выберет кнопку «Вычисления», то он автоматически перейдет на «Страницу вычислений», и одновременно с ней откроется форма очистки полей (рис. 6.4). В обоих случаях форма выбора станет невидимой и недоступной для использования до момента закрытия формы справки или формы вычислений.

Рисунок 6.4 − Страница вычислений (главная форма программы)

На главной форме отображаются кнопки с путем ко всем операциям, выполняемым программой. Для перехода к необходимой операции, следует нажать на копку с ее названием на форме или же выбрать пункт «Операции – «название операции» в файловом меню (рис. 6.4).

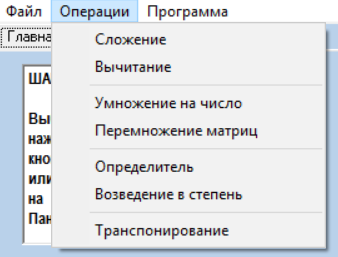
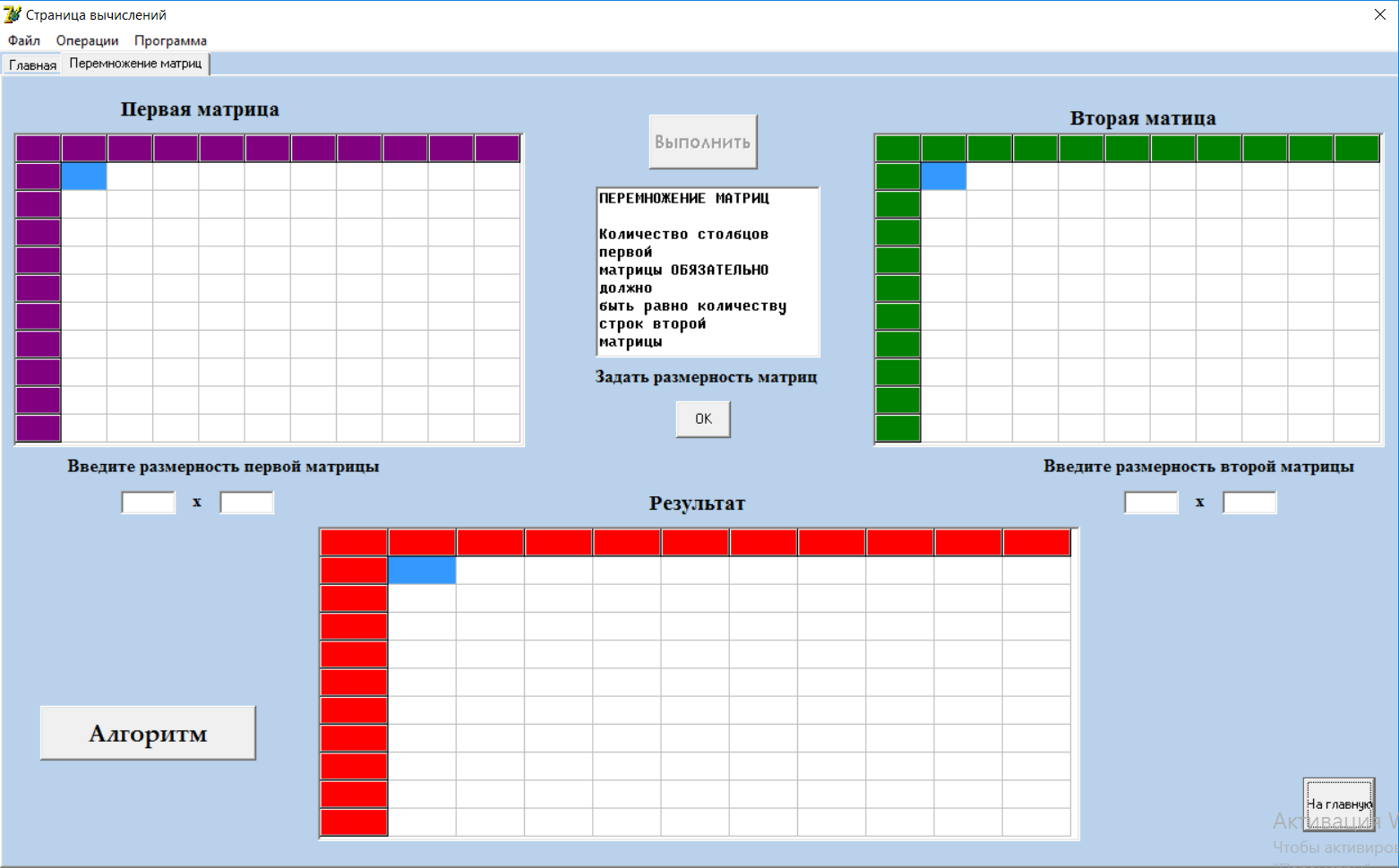


Рисунок 6.5 − Переход к операции через файловое меню

Также с главной формы через файловое меню можно перейти к текстовой справке путем выбора пунктов «Программа - Справка», открыть страницу сведений о программе и ее разработчике через «Программа – О программе», а так же найти кнопку «Выполнить», которую можно использовать как кнопку команды для начала проведения операции, во вкладке «Программа - Выполнить». Также с этой страницы возможен вызов программного калькулятора (рис. 6.7).

Рассмотрим работу с вкладками вычислений при помощи вкладки «Перемножение матриц» (рис. 6.5).

Рисунок 6.6 − Вкладка перемножения матриц



Как видно из рисунка 6.6, стандартная вкладка представляет собой страницу с несколькими полями для внесения матриц, полем результата, полями для внесения размерности матриц, кнопка утверждения размерности матриц, кнопка выполнения операции, поля с теорией по конкретной операции, а также кнопками открытия алгоритма и возврата на главную страницу. Использование приложения упрощается наличием подписей-подсказок.

Из рисунка 6.5 можно наблюдать, что изначально кнопка «Выполнить» является недоступной для использования. В таком состоянии она будет находится, пока не будет нажата кнопка утверждения размерности матриц «ОК». При ее нажатии приложение выделяет память под конкретное количество элементов массивов, заранее введенное в пустые поля под подписью «Введите размерность матрицы». Всякий раз, когда будет меняться одно из пустых полей, кнопка «Выполнить» будет становиться недоступной для использования до нажатия кнопки «ОК». Стоит отметить, что и с кнопкой файлового меню «Программа - Выполнить» будет происходить то же самое. Таким образом, она является полным аналогом кнопки «Выполнить» для каждой вкладки страницы вычислений.

После задания размеров в поля матриц необходимо внести сами элементы матрицы, после чего нажать на «Выполнить» и в поле «Результат» появится результат вычисления, который будет занесен во временный файл, созданный в папке программы.

Путем нажатия клавиши «Алгоритм» будет показан алгоритм конкретной операции. В случае клика на кнопку «На главную», пользователь вернется к форме, показанной на рисунке 6.4.

На рисунке 6.7 показана панель, всплывающая при нажатии на кнопку «Файл» файлового меню программы. Здесь можно найти кнопку «Новый», после нажатия на которую автоматически будут очищены поля текущей (открытой) вкладки вычислений, «Открыть», после клика по которой появится диалоговое окно для загрузки данных в программу из файла, а также «Сохранить», при нажатии на которую опять же появится диалоговое окно, но уже для сохранения результата вычисления в некоторый файл на компьютере или съемном носителе.

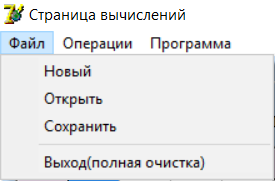


Рисунок 6.7 − Панель «Файл» файлового меню программы

При нажатии на кнопку «Выход (полная очистка)» страница вычислений автоматически закроется, при этом будут очищены все поля каждой вкладки этой странницы.

Программный калькулятор, показанный на рисунке 6.8, представляет собой калькулятор, способный выполнять стандартные математические операции (сложение, вычитание, умножение, деление, смена знака). Также в него внесена функция добавления числа из поля ввода в список (историю) вычислений. Для этого надо нажать на кнопку «А» или же воспользоваться сочетанием горячих клавиш «alt+A». При этом произойдет добавление этого элемента в поле в правой части калькулятора и во временный файл на компьютере.

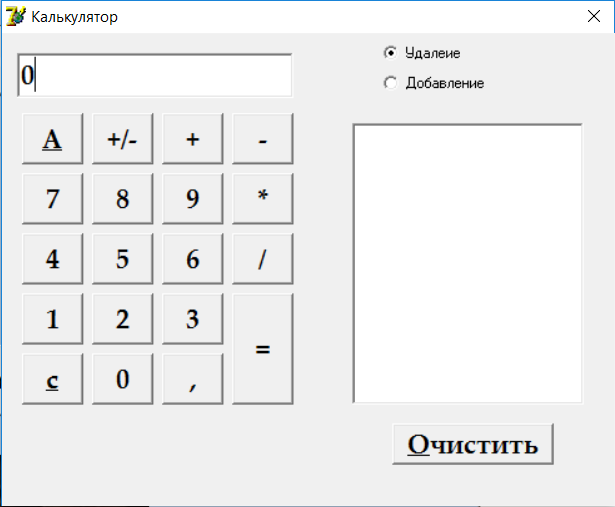


Рисунок 6.8 − Программный калькулятор

В приложении предусмотрена возможность удаления элемента из списка. Для этого необходимо установиь положение выбора в графу «Удаление» и кликнуть по нужному элементу, после чего он удалится как из поля, так и из файла на компьютере. Кроме всего прочего возможным остается быстрое удаление всего списка путем нажатия клавиши «Очистить».

Клавиша «+/-» служит в калькуляторе для смены знака с плюса на минус и наоборот. Клавиша «C» для сброса текущих значений и придания полю ввода нулевого значения.

Если пользователь желает добавить в поле вычислений элемент из истории, то он должен установить выбор на графу «Добавление» и кликнуть по элементу в поле истоирии.

При открытии страницы вычислений происходит автоматическое открытие формы очищения полей всех вкладок (рис. 6.9), которая не требует выхода из приложения или закрытия формы вычислений. Она будет активна пока активно окно операций. Форма представлена единственной кнопкой, при нажатии на которую и происходят вышеуказанные действия.

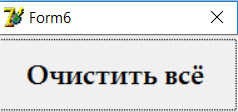


Рисунок 6.9 − Форма очищения полей

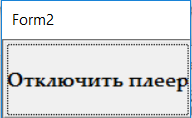


Рисунок 6.10 − Форма отключения плеера

**ЗАКЛЮЧЕНИЕ**

В результате работы над курсовым проектом было создано исправно работающее приложение для выполнения операций над матрицами, которое может пригодиться в учебе и работе людям, чья жизнь связана с математикой.

Разработка приложения включала в себя решение множества задач и проблем, как итог было изучено большое количество матричных калькуляторов, проведен их анализ, и сформированы оптимальные требования для приложений подобного рода.

Далее были изучены некоторые возможности создания приложений в Delphi 7 и формирование конкретных функциональных требований к программе на основе возможностей языка.

Затем были разработаны структуры данных, разработана примерная архитектура приложения. Далее были детализированы все функции.

Программа была отлажена и протестирована сначала разработчиком, а затем несколько раз обычными пользователями. После испытаний были внесены корректировки в интерфейсе, работе некоторых функций.

Пройдя все вышеперечисленные этапы на выходе получилась корректно работающее программное средство для проведения операций над матрицами.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] **Сайт для онлайн-изучения математики “OnlineMSchool”** [Электронный ресурс]**.** – Электронные данные. – Режим доступа: http://ru.onlinemschool.com/math/assistance/matrix/transpose/ Дата доступа: 24.03.17

[2]Официальный сайт онлайн-калькулятора “WolframAlpha [Электронный ресурс]**.** – Электронные данные. – Режим доступа: <http://www.wolframalpha.com/examples/Matrices.html> Дата доступа: 26.03.17

[3]Softportal. Программы. “MathCad15” [Электронный ресурс]**.** – Электронные данные. – Режим доступа: http://www.ptc.com/engineering-math-software/mathcad/ Дата доступа: 27.03.17

[4] Softportal. Программы. “MatrixCalculatorPro” [Электронный ресурс]**.** – Электронные данные. – Режим доступа: http://freesoft.ru/matrix\_calculator\_pro/ Дата доступа: 29.03.16

[5] Справочник – “Основы Delphi” [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://delphibasics.ru/>.

[6] ГОСТ 19.701–90 (ИСО 5807–85) [Текст]. – Единая система программной документации: Сб. ГОСТов. - М.: Стандартинформ, 2005 с.

[7] Письменный Д.Т. Конспекты лекций по математике. Полный курс: учебное пособие для изучающих высшую математику. / Письменный Д.Т. – Москва: АЙРИС ПРЕСС, 2006. – 241 с. : ил.

[8] Серебряная Л.В., Марина И.М. Структуры и алгоритмы обработки данных: учебно-методическое пособие для студ. спец. «Программное обеспечение информационных технологий» всех форм обуч. / Серебряная Л.В., Марина И.М. – Минск: БГУИР, 2012. – 49 с. : ил.

**ПРИЛОЖЕНИЕ. ИСХОДНЫЙ КОД ПРОГРАММЫ**

**program Matrix\_Calculator\_Pro2;**

uses

Forms,

uHi in 'uHi.pas' {fmHi},

uChoise in 'uChoise.pas' {fmChoise},

uOperations in 'uOperations.pas' {fmOper},

uAboutProgramm in 'uAboutProgramm.pas' {fmAboutProgramm},

uCalculator in 'uCalculator.pas' {fmCalculator},

uOpenUChoise in 'uOpenUChoise.pas' {Form6},

uInstruction in 'uInstruction.pas' {fmInstruction},

uActions in 'uActions.pas',

uMultMatr in 'uMultMatr.pas' {fmAlgoritm},

uSound in 'uSound.pas' {Form1},

uPlayerOff in 'uPlayerOff.pas' {Form2};

{$R \*.res}

begin

With Application do

begin

Initialize;

CreateForm(TfmHi, fmHi);

CreateForm(TfmChoise, fmChoise);

CreateForm(TfmOper, fmOper);

CreateForm(TfmAboutProgramm, fmAboutProgramm);

CreateForm(TfmCalculator, fmCalculator);

CreateForm(TForm6, Form6);

Run;

end;

**end.**

**Unit uAboutProgramm**

**unit uAboutProgramm;**

interface

uses

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

Dialogs, StdCtrls, jpeg, ExtCtrls;

type

TfmAboutProgramm = class(TForm)

Image1: TImage;

Label1: TLabel;

memo1: TMemo;

private

{ Private declarations }

public

{ Public declarations }

end;

var

fmAboutProgramm: TfmAboutProgramm;

implementation

{$R \*.dfm}

**end.**

**Unit uActions**

**unit uActions;**

interface

uses

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

Dialogs, ComCtrls, Menus, StdCtrls, ActnList, XPStyleActnCtrls, ActnMan,

Grids,

uOperations, uHi, uSound;

procedure SaveTable(Table : TStringGrid; str : string);

procedure SaveEdit(Edit : TEdit);

procedure CheckString(var Table : TStringGrid; i , j :integer; var flag : boolean);

procedure CheckEdit(var Edit : TEdit; var flag : boolean);

procedure SetEdit(Edit : TEdit; var size : integer);

procedure Clean(Table : TStringGrid);

procedure Add;

procedure Determinate;

function Det(arr : TMatrix; r : integer; c : TColumn; const n : integer) : real;

procedure Sub;

procedure MultMatrix;

procedure MultNumb;

procedure Pow;

procedure Transpose;

procedure Music(s : string);

implementation

procedure Music(s : string);

begin

fmOper.MediaPlayer1.FileName := way + s;

fmOper.MediaPlayer1.Open;

fmOper.MediaPlayer1.Play;

end;

**//процедура сохранения информации из таблицы в файл**

procedure SaveTable(Table: TStringGrid; str : string);

var

i, j: integer;

begin

if n = 0 then

assignfile(f, way + 'Результаты вычислений.txt')

else assignfile(f, fmOper.SaveDialog1.FileName);

append(f);

write(f, 'Результат ' + str + ':');

with Table do

for i:=0 to RowCount-1 do

begin

for j:=0 to ColCount-1 do

write(f, Table.cells[j, i] + ' ');

writeln(f, #13 + #13);

end;

closefile(f);

end;

**//процедура сохранения данныз из поля в файл**

procedure SaveEdit(Edit : TEdit);

begin

if n = 0 then

assignfile(f, way + 'Результаты вычислений.txt')

else assignfile(f, fmOper.SaveDialog1.FileName);

append(f);

writeln(f, 'Результат нахождения определителя:');

writeln(f, Edit.Text);

writeln(f, #13 + #13);

closefile(f);

end;

**//процедуоа провери ячеек таблицы на корректность**

procedure CheckString(var Table : TStringGrid; i, j: integer; var flag : boolean);

var

k, x : integer;

curr : string;

begin

flag := true;

x := 0;

curr := Table.Cells[j+1, i+1];

for k := 1 to length(curr) do

begin

if curr[k] = ',' then inc(x);

if curr[k] = '.' then curr[k] := ',';

if not(curr[k] in ['0','1','2','3','4','5','6','7','8','9', ',','-']) or

(length(curr) = 1) and (curr[1] = '-') or

(length(curr) > 1) and (k >= 2) and (curr[k] = '-') or

(curr[1] = ',') or (x > 1) then

begin

messagebox(fmoper.Handle, pchar('Некорректные данные! Результаты так же неверны!'), pchar('Ошибка'), mb\_ok+mb\_iconerror);

flag := false;

if (sound = true) and (player = false) then

Music('Предупреждения\Ошибки в расчетах.mp3');

break;

end;

end;

Table.Cells[j+1, i+1] := curr;

end;

**//процедуоа провери поля на корректность**

procedure CheckEdit(var Edit : TEdit; var flag : boolean);

var

k, x : integer;

curr : string;

begin

flag := true;

x := 0;

curr := Edit.Text;

for k := 1 to length(curr) do

begin

if curr[k] = ',' then inc(x);

if curr[k] = '.' then curr[k] := ',';

if not(curr[k] in ['0','1','2','3','4','5','6','7','8','9', ',','-']) or

(length(curr) = 1) and (curr[1] = '-') or

(length(curr) > 1) and (k >= 2) and (curr[k] = '-') or

(curr[1] = ',') or (x > 1) then

begin

messagebox(fmoper.Handle, pchar('Некорректные данные! Результаты так же неверны!'), pchar('Ошибка'), mb\_ok+mb\_iconerror);

flag := false;

if (sound = true) and (player = false) then

Music('Предупреждения\Плохие данные.mp3');

break;

end;

end;

Edit.Text := curr;

end;

**//процедура очистки всех ячеек таблицы**

procedure Clean(Table : TStringGrid);

var

i :integer;

begin

with Table do

for i := 1 to ColCount-1 do

Cols[i].Clear;

end;

**//процедура преобразования информации из поля в нужный тип**

procedure SetEdit(Edit : TEdit; var size : integer);

begin

if (Edit.Text = '') or (strtofloat(Edit.Text) < 0) then

size := 0

else size := trunc(strtofloat(Edit.Text));

if size > 10 then size := 10;

Edit.Text := inttostr(size);

end;

**///////////////////////////////////////////////////////////////////////////////**

**// раздел описания процедур и функций над матрицами(1)**

**//процедура сложения**

procedure Add;

var

i, j, k:integer;

arr1, arr2, arr3: TMatrix;

flag : boolean;

begin

flag := true;

setlength(arr1, size1, size2);

setlength(arr2, size1, size2);

setlength(arr3, size1, size2);

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

CheckString(fmOper.strFirstAdd, i, j, flag);

if flag = false then exit;

if fmOper.strFirstAdd.Cells[j+1, i+1] = '' then

begin

arr1[i, j] := 0;

fmOper.strFirstAdd.Cells[j+1, i+1] := '0';

end

else

arr1[i, j] := strtofloat(fmOper.strFirstAdd.Cells[j+1, i+1]);

if iden = 0 then

begin

CheckString(fmOper.strSecondAdd, i, j, flag);

if flag = false then exit;

if fmOper.strSecondAdd.Cells[j+1, i+1] = '' then

begin

arr2[i, j] := 0;

fmOper.strSecondAdd.Cells[j+1, i+1] := '0';

end

else

arr2[i, j] := strtofloat(fmOper.strSecondAdd.Cells[j+1, i+1]);

end

else arr2[i, j] := numb;

arr3[i, j] := arr1[i, j] + arr2[i, j];

fmOper.strResultAdd.Cells[j+1, i+1] := floattostr(arr3[i, j]);

end;

if (sound = true) and (player = false) then

Music('Предупреждения\Всё хорошо.mp3');

SaveTable(fmOper.strResultAdd, 'сложения');

end;

**//процедура вычитания**

procedure Sub;

var

i, j:integer;

arr1, arr2, arr3: TMatrix;

flag : boolean;

begin

flag := true;

setlength(arr1, size1, size2);

setlength(arr2, size1, size2);

setlength(arr3, size1, size2);

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

CheckString(fmOper.strFirstSub, i, j, flag);

if flag = false then exit;

if fmOper.strFirstSub.Cells[j+1, i+1] = '' then

begin

arr1[i, j] := 0;

fmOper.strFirstSub.Cells[j+1, i+1] := '0';

end

else

arr1[i, j] := strtofloat(fmOper.strFirstSub.Cells[j+1, i+1]);

if iden = 0 then

begin

CheckString(fmOper.strSecondSub, i, j, flag);

if flag = false then exit;

if fmOper.strSecondSub.Cells[j+1, i+1] = '' then

begin

arr2[i, j] := 0;

fmOper.strSecondSub.Cells[j+1, i+1] := '0';

end

else

arr2[i, j] := strtofloat(fmOper.strSecondSub.Cells[j+1, i+1])

end

else arr2[i, j] := numb ;

if sign = 1 then

arr3[i, j] := arr1[i, j] - arr2[i, j]

else arr3[i, j] := -arr1[i, j] + arr2[i, j];

fmOper.strResultSub.Cells[j+1, i+1] := floattostr(arr3[i, j]);

end;

if (sound = true) and (player = false) then

Music('Предупреждения\Всё хорошо.mp3');

SaveTable(fmOper.strResultSub, 'вычитания');

end;

**//процедура умножения матрицы на число**

procedure MultNumb;

var

i, j: integer;

arr1, arr2: TMatrix;

flag : boolean;

begin

flag := true;

setlength(arr1, size1, size2);

setlength(arr2, size1, size2);

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

CheckString(fmOper.strMultNumb, i, j, flag);

if flag = false then exit;

if fmOper.strMultNumb.Cells[j+1, i+1] = '' then

begin

arr1[i, j] := 0;

fmOper.strMultNumb.Cells[j+1, i+1] := '0';

end

else

arr1[i, j] := strtofloat(fmOper.strMultNumb.Cells[j+1, i+1]);

arr2[i, j] := arr1[i, j] \* numb ;

fmOper.strResultMultNumb.Cells[j+1, i+1] := floattostrf(arr2[i, j], ffGeneral, 8, 4);

end;

if (sound = true) and (player = false) then

Music('Предупреждения\Всё хорошо.mp3');

SaveTable(fmOper.strResultMultNumb, 'умножения на число');

end;

**//процедура возведения матрицы в степень**

procedure Pow;

var

ia, ib, ja, jb, k, i, j: integer;

s : real;

arr1, arr2, arr3: TMatrix;

flag : boolean;

begin

flag := true;

setlength(arr1, size1 ,size1);

setlength(arr2, size1 ,size1);

setlength(arr3, size1 ,size1);

for i:=0 to size1-1 do

for j:=0 to size1-1 do

begin

CheckString(fmOper.strPow, i, j, flag);

if flag = false then exit;

if fmOper.strPow.Cells[j+1, i+1] = '' then

begin

arr1[i, j] := 0;

fmOper.strPow.Cells[j+1, i+1] := '0';

end

else arr1[i, j] := strtofloat(fmOper.strPow.cells[j+1, i+1]);

arr2[i, j] := arr1[i, j];

end;

k:=1;

while k <= numb1-1 do

begin

for ia:=0 to size1-1 do

for jb:=0 to size1-1 do

begin

s := 0;

ja := 0;

ib := 0;

while (ja <= size1-1) do

begin

s := s + arr1[iA, jA] \* arr2[iB,jB];

inc(ja);

inc(ib);

end;

arr3[ia,jb] := s;

end;

inc(k);

for i:=0 to size1-1 do

for j:=0 to size1-1 do

arr2[i, j] := arr3[i, j];

end;

for i := 0 to size1-1 do

for j := 0 to size1-1 do

fmOper.strResultPow.cells[j+1, i+1] := floattostrf(arr3[i, j], ffGeneral, 8, 4);

if (sound = true) and (player = false) then

Music('Предупреждения\Всё хорошо.mp3');

SaveTable(fmOper.strResultPow, 'возведения в степень');

end;

**//функция вычисления определителя**

function Det(arr : TMatrix; r : integer; c: TColumn; const n : integer) : real;

var

i : integer;

sum : real;

minusflag : boolean;

begin

sum := 0;

minusflag := false;

if r = n then

begin

for i := 0 to n-1 do

if not(i in c) then

sum := arr [i, r-1];

end

else for i := 0 to n-1 do

if not(i in c) then

begin

include(c, i);

if not minusflag then

sum := sum + arr[i, r-1] \* Det(arr, r+1, c, n)

else sum := sum - arr[i, r-1] \* Det(arr, r+1, c, n);

exclude(c, i);

minusflag := not minusflag;

end;

result := sum;

end;

**//процедура нахождения определителя**

procedure Determinate;

var

arr : TMatrix;

i, j : integer;

flag : boolean;

begin

flag := true;

setlength(arr, size1, size1);

for i := 0 to size1-1 do

for j := 0 to size1-1 do

begin

CheckString(fmOper.strDet, i, j, flag);

if flag = false then exit;

if fmOper.strDet.Cells[j+1, i+1] = '' then

begin

arr[i, j] := 0;

fmOper.strDet.Cells[j+1, i+1] := '0';

end

else

arr[i, j] := strtofloat(fmOper.strDet.Cells[j+1, i+1]);

end;

fmOper.edResultDet.Text := floattostr(det(arr, 1, [], size1));

if (sound = true) and (player = false) then

Music('Предупреждения\Всё хорошо.mp3');

SaveEdit(fmOper.edResultDet);

end;

**//процедура перемножения матриц**

procedure MultMatrix;

var

arr1, arr2, arr3 : TMatrix;

iA, jA, iB, jB, i, j : integer;

sum : real;

flag : boolean;

begin

flag := true;

setlength(arr1, size1, size2);

setlength(arr2, size1mult, size2mult);

setlength(arr3, sizeresmult1, sizeresmult2);

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

CheckString(fmOper.strFirstMultMatr, i, j, flag);

if flag = false then exit;

if fmOper.strFirstMultMatr.cells[j+1, i+1] = '' then

begin

arr1[i, j] := 0;

fmOper.strFirstMultMatr.cells[j+1, i+1] := '0';

end

else arr1[i, j] := strtofloat(fmOper.strFirstMultMatr.Cells[j+1, i+1]);

end;

for i := 0 to size1mult-1 do

for j := 0 to size2mult-1 do

begin

CheckString(fmOper.strSecondMultMatr, i, j, flag);

if flag = false then exit;

if fmOper.strSecondMultMatr.cells[j+1, i+1] = '' then

begin

arr2[i, j] := 0;

fmOper.strSecondMultMatr.cells[j+1, i+1] := '0';

end

else arr2[i, j] := strtofloat(fmOper.strSecondMultMatr.Cells[j+1, i+1]);

end;

for iA := 0 to size1-1 do

for jB := 0 to size2mult-1 do

begin

sum := 0;

jA := 0;

iB := 0;

while (jA <= size2-1) and (iB <= size1mult-1) do

begin

sum := sum + arr1[iA, jA] \* arr2[iB, jB];

inc(jA);

inc(iB);

end;

arr3[iA, jB] := sum;

end;

for i := 0 to sizeresmult1-1 do

for j := 0 to sizeresmult2-1 do

fmOper.strResultMultMatr.Cells[j+1, i+1] := floattostrf(arr3[i, j], ffGeneral, 8, 4);

if (sound = true) and (player = false) then

Music('Предупреждения\Всё хорошо.mp3');

SaveTable(fmOper.strResultMultMatr, 'перемножения матриц');

end;

**//процедура транспонирования матрицы**

procedure Transpose;

var

i, j : integer;

k : real;

arr1, arr2 : TMatrix;

flag : boolean;

begin

flag := true;

setlength(arr1, size1, size2);

setlength(arr2, size2, size1);

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

CheckString(fmOper.strTranspose, i, j, flag);

if flag = false then exit;

if fmOper.strTranspose.Cells[j+1, i+1] = '' then

begin

arr1[i, j] := 0;

fmOper.strTranspose.Cells[j+1, i+1] := '0';

end

else

arr1[i, j] := strtofloat(fmOper.strTranspose.Cells[j+1, i+1]);

end;

for i := 0 to size2-1 do

for j := 0 to size1-1 do

arr2[i, j] := arr1[j, i];

for i := 0 to size2-1 do

for j := 0 to size1-1 do

fmOper.strResultTranspose.Cells[j+1, i+1] := floattostr(arr2[i, j]);

if (sound = true) and (player = false) then

Music('Предупреждения\Всё хорошо.mp3');

SaveTable(fmOper.strResultTranspose, 'транспонирования');

end;

**end.**

**Unit uCalculator**

**unit uCalculator;**

interface

uses

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

Dialogs, StdCtrls , uHi;

type

TfmCalculator = class(TForm)

EPoleEnter: TEdit;

B1: TButton;

B2: TButton;

B3: TButton;

B4: TButton;

B5: TButton;

B6: TButton;

B7: TButton;

B8: TButton;

B9: TButton;

b0: TButton;

BC: TButton;

BPoint: TButton;

BEnter: TButton;

BDiv: TButton;

BMultiply: TButton;

BMinus: TButton;

BPlus: TButton;

BSign: TButton;

BAddList: TButton;

ListBox1: TListBox;

BDelete: TButton;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

procedure b0Click(Sender: TObject);

procedure B1Click(Sender: TObject);

procedure B2Click(Sender: TObject);

procedure B3Click(Sender: TObject);

procedure B4Click(Sender: TObject);

procedure B5Click(Sender: TObject);

procedure B6Click(Sender: TObject);

procedure B7Click(Sender: TObject);

procedure B8Click(Sender: TObject);

procedure B9Click(Sender: TObject);

procedure BPointClick(Sender: TObject);

procedure BSignClick(Sender: TObject);

procedure BAddListClick(Sender: TObject);

procedure BPlusClick(Sender: TObject);

procedure BMinusClick(Sender: TObject);

procedure BMultiplyClick(Sender: TObject);

procedure BDivClick(Sender: TObject);

procedure BEnterClick(Sender: TObject);

procedure BCClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure BDeleteClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure ListBox1Click(Sender: TObject);

procedure FormShow(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fmCalculator: TfmCalculator;

implementation

uses uOperations, uSound, uActions;

{$R \*.dfm}

type

TOperation=(oDo,oPlus,oMinus,oMultiply,oDiv);

var

accum: real;

Operation: TOperation;

f, k: byte;

**//процедура обработки введенных чисел-символов и их преобразования в числа**

procedure NumbShow(symb : char);

begin

with fmCalculator do

begin

if f = 0 then

begin

EPoleEnter.Text := symb;

f := 1;

end

else

begin

if k = 0 then

if EPoleEnter.Text <> symb then

EPoleEnter.Text := EPoleEnter.Text + symb;

if k <> 0 then

EPoleEnter.Text := EPoleEnter.Text + symb;

end;

end;

end;

**//процедура проведения операций**

procedure DoOperation;

var

numb: real;

s: string;

l : integer;

begin

s := fmCalculator.EPoleEnter.Text;

l := 0;

numb := StrToFloat(s);

case Operation of

oDo: begin

accum := numb;

inc(l);

end;

oPlus: accum := accum + numb;

oMinus: accum := accum - numb;

oMultiply: accum := accum \* numb;

oDiv: begin

if numb = 0 then

begin

MessageBox(fmCalculator.handle,PChar('Нельзя делить на ноль!'),PChar('Ошибка'),(MB\_OK+MB\_ICONERROR));

if (sound = true) and (player = false) then

Music('Предупреждения\На ноль делить нельзя.mp3');

end

else accum := accum / numb;

end;

end;

fmCalculator.EPoleEnter.Text := FloatToStr(accum)

end;

**//процедура направления на операцию или продолжения ввода**

procedure Doing(Oper : TOperation);

begin

if f = 0

then Operation:=oper

else

begin

DoOperation;

Operation:=oper;

f:=0;

end;

end;

procedure TfmCalculator.b0Click(Sender: TObject);

begin

k := 0;

NumbShow('0');

end;

procedure TfmCalculator.B1Click(Sender: TObject);

begin

k := 1;

NumbShow('1');

end;

procedure TfmCalculator.B2Click(Sender: TObject);

begin

k := 1;

NumbShow('2');

end;

procedure TfmCalculator.B3Click(Sender: TObject);

begin

k := 1;

NumbShow('3');

end;

procedure TfmCalculator.B4Click(Sender: TObject);

begin

k := 1;

NumbShow('4');

end;

procedure TfmCalculator.B5Click(Sender: TObject);

begin

k := 1;

NumbShow('5');

end;

procedure TfmCalculator.B6Click(Sender: TObject);

begin

k := 1;

NumbShow('6');

end;

procedure TfmCalculator.B7Click(Sender: TObject);

begin

k := 1;

NumbShow('7');

end;

procedure TfmCalculator.B8Click(Sender: TObject);

begin

k := 1;

NumbShow('8');

end;

procedure TfmCalculator.B9Click(Sender: TObject);

begin

k := 1;

NumbShow('9');

end;

procedure TfmCalculator.BPointClick(Sender: TObject);

begin

if fmCalculator.EPoleEnter.Text = '0' then

begin

fmCalculator.EPoleEnter.Text := '0,';

f := 1;

end;

if Pos(',',EPoleEnter.Text) = 0 then

fmCalculator.EPoleEnter.Text := fmCalculator.EPoleEnter.Text + ',';

end;

**//процедура замены знака**

procedure TfmCalculator.BSignClick(Sender: TObject);

const

s1 = '-';

var

s: string;

begin

s := fmCalculator.EPoleEnter.Text;

if s[1] = s1 then

begin

delete(s, 1, 1);

fmCalculator.EPoleEnter.Text := s;

end

else

fmCalculator.EPoleEnter.Text := s1 + fmCalculator.EPoleEnter.Text;

end;

**//процедура добавления результатов в список**

procedure TfmCalculator.BAddListClick(Sender: TObject);

begin

if fmCalculator.EPoleEnter.Text = '' then

ListBox1.Items.Add('0')

else ListBox1.Items.Add(fmCalculator.EPoleEnter.Text);

fmCalculator.ListBox1.Items.SaveToFile(way + 'Занесение в файл.txt');

end;

procedure TfmCalculator.BPlusClick(Sender: TObject);

begin

doing(oplus);

end;

procedure TfmCalculator.BMinusClick(Sender: TObject);

begin

doing(ominus);

end;

procedure TfmCalculator.BMultiplyClick(Sender: TObject);

begin

doing(omultiply);

end;

procedure TfmCalculator.BDivClick(Sender: TObject);

begin

doing(odiv);

end;

procedure TfmCalculator.BEnterClick(Sender: TObject);

begin

doing(odo);

end;

procedure TfmCalculator.BCClick(Sender: TObject);

begin

EPoleEnter.Text := '0';

accum := 0;

Operation := oDo;

f := 0;

end;

procedure TfmCalculator.FormCreate(Sender: TObject);

begin

Operation:=oDo;

EPoleEnter.ReadOnly:=true;

end;

**//процедура блокировки нечисловых клавиш клавиатуры**

procedure TfmCalculator.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

case Key of

ord('0'),96: fmCalculator.b0.Click;

ord('1'),97: fmCalculator.B1.Click;

ord('2'),98: fmCalculator.B2.Click;

ord('3'),99: fmCalculator.B3.Click;

ord('4'),100: fmCalculator.B4.Click;

ord('5'),101: fmCalculator.B5.Click;

ord('6'),102: fmCalculator.B6.Click;

ord('7'),103: fmCalculator.B7.Click;

ord('8'),104: fmCalculator.B8.Click;

ord('9'),105: fmCalculator.B9.Click;

106: fmCalculator.BMultiply.Click;

107: fmCalculator.BPlus.Click;

109: fmCalculator.BMinus.Click;

110: fmCalculator.BPoint.Click;

111: fmCalculator.BDiv.Click;

13: fmCalculator.BEnter.Click;

end;

end;

procedure TfmCalculator.BDeleteClick(Sender: TObject);

begin

ListBox1.Items.Clear;

end;

procedure TfmCalculator.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

ListBox1.Items.Clear;

fmOper.btCalc.Enabled := true;

end;

**//процедура удаления из временного файла и добавления в поле**

procedure TfmCalculator.ListBox1Click(Sender: TObject);

var

h : integer;

begin

h := fmCalculator.ListBox1.ItemIndex;

if RadioButton1.Checked = true then

begin

fmCalculator.ListBox1.Items.Delete(h);

fmCalculator.ListBox1.Items.SaveToFile(way + 'Занесение в файл.txt');

end;

if RadioButton2.Checked = true then

begin

EPoleEnter.Text := ListBox1.Items[h];

end;

end;

procedure TfmCalculator.FormShow(Sender: TObject);

begin

EPoleEnter.Text := '0';

RadioButton1.Checked := true;

end;

**end.**

**Unit uChoise**

**unit uChoise;**

interface

uses

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

Dialogs, jpeg, ExtCtrls, StdCtrls;

type

TfmChoise = class(TForm)

imChoise: TImage;

butCount: TButton;

lbChoise: TLabel;

butInst: TButton;

procedure butCountClick(Sender: TObject);

procedure butInstClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fmChoise: TfmChoise;

implementation

uses uOperations, uOpenUChoise, uInstruction, uHi;

{$R \*.dfm}

procedure TfmChoise.butCountClick(Sender: TObject);

begin

fmOper.show;

Form6.Show;

fmChoise.Enabled := false;

fmChoise.Visible := false;

end;

procedure TfmChoise.butInstClick(Sender: TObject);

begin

if not assigned(fmInstruction) then

fmInstruction := tfmInstruction.create(self);

fmInstruction.show;

fmChoise.Enabled := false;

fmChoise.Visible := false;

end;

procedure TfmChoise.FormClose(Sender: TObject; var Action: TCloseAction);

begin

fmHi.close;

end;

end.

**Unit uHi**

**unit uHi;**

**interface**

uses

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

Dialogs, StdCtrls, jpeg, ExtCtrls, Buttons, MPlayer;

type

TfmHi = class(TForm)

imNeo: TImage;

lbMCalc: TLabel;

butStart: TButton;

procedure butStartClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fmHi: TfmHi;

f : textfile;

way : string;

implementation

uses uChoise, uSound;

{$R \*.dfm}

procedure TfmHi.butStartClick(Sender: TObject);

begin

if not assigned(form1) then

form1 := tform1.create(self);

form1.show;

fmHi.Enabled := false;

fmHi.Visible := false;

end;

**//процедура создания временных файлов и запоминания абсолютного пути к проекту**

procedure TfmHi.FormCreate(Sender: TObject);

begin

assignfile(f, ExtractFilePath(ExpandFileName('Project1.exe')) + 'Ðåçóëüòàòû âû÷èñëåíèé.txt');

way := ExtractFilePath(ExpandFileName('Project1.exe'));

rewrite(f);

closefile(f);

assignfile(f, way + 'Çàíåñåíèå â ôàéë.txt');

rewrite(f);

closefile(f);

end;

procedure TfmHi.FormClose(Sender: TObject; var Action: TCloseAction);

begin

if player = true then

dispose(tracklist);

end;

**end.**

**Unit uInstruction**

**unit uInstruction;**

interface

uses

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

Dialogs, StdCtrls, ComCtrls, uHi;

type

TfmInstruction = class(TForm)

rchInstruction: TRichEdit;

procedure FormShow(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

fmInstruction: TfmInstruction;

implementation

uses uChoise;

{$R \*.dfm}

procedure TfmInstruction.FormShow(Sender: TObject);

begin

fmInstruction.rchInstruction.Lines.LoadFromFile( way + 'Èíñòðóêöèÿ ïîëüçîâàòåëÿ.txt');

end;

procedure TfmInstruction.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

fmChoise.Enabled := true;

fmChoise.Visible := true;

end;

**end.**

**Unit uMultMatr**

**unit uMultMatr;**

interface

uses

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

Dialogs, jpeg, ExtCtrls;

type

TfmAlgoritm = class(TForm)

Image1: TImage;

private

{ Private declarations }

public

{ Public declarations }

end;

var

fmAlgoritm: TfmAlgoritm;

implementation

{$R \*.dfm}

**end.**

**Unit uOpenUChoise**

**unit uOpenUChoise;**

interface

uses

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

Dialogs, StdCtrls;

type

TForm6 = class(TForm)

Button1: TButton;

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form6: TForm6;

implementation

uses uOperations;

{$R \*.dfm}

procedure TForm6.Button1Click(Sender: TObject);

begin

cleanall;

end;

end.

Unit uOperations

unit uOperations;

interface

uses

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

Dialogs, ComCtrls, Menus, StdCtrls, ActnList, XPStyleActnCtrls, ActnMan,

Grids, ExtCtrls, jpeg, MPlayer;

type

TMatrix = array of array of real;

TColumn = set of 1..10;

TMatr = array [1..10, 1..10] of real;

TfmOper = class(TForm)

fmOperMenu: TMainMenu;

PageControl1: TPageControl;

tabMain: TTabSheet;

btCalc: TButton;

tabAdd: TTabSheet;

tabSub: TTabSheet;

mFile: TMenuItem;

mNew: TMenuItem;

mOpen: TMenuItem;

mSave: TMenuItem;

N6: TMenuItem;

mExit: TMenuItem;

mOperations: TMenuItem;

mAdd: TMenuItem;

mSub: TMenuItem;

N11: TMenuItem;

mMultNumb: TMenuItem;

mMultMatr: TMenuItem;

N14: TMenuItem;

mDet: TMenuItem;

mPow: TMenuItem;

mProgramm: TMenuItem;

mInst: TMenuItem;

mAboutProgramm: TMenuItem;

btAdd: TButton;

btSub: TButton;

btDet: TButton;

btMultNumb: TButton;

btMultMatr: TButton;

btPow: TButton;

tabMultNumb: TTabSheet;

tabMultMatr: TTabSheet;

tabPow: TTabSheet;

tabDet: TTabSheet;

btBackToMain: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Button5: TButton;

btAddNumb: TButton;

btAddMatr: TButton;

strFirstAdd: TStringGrid;

strSecondAdd: TStringGrid;

strResultAdd: TStringGrid;

lbResultAdd: TLabel;

lbFirstAdd: TLabel;

lbSecondAdd: TLabel;

strFirstSub: TStringGrid;

lbResultSub: TLabel;

lbFirstSub: TLabel;

lbSecondSub: TLabel;

strSecondSub: TStringGrid;

strResultSub: TStringGrid;

btSubMatr: TButton;

btSubNumb: TButton;

strMultNumb: TStringGrid;

lbResultMultNumb: TLabel;

strResultMultNumb: TStringGrid;

strFirstMultMatr: TStringGrid;

lbResultMutMatr: TLabel;

lbFirstMultMatr: TLabel;

lbSecondMultMatr: TLabel;

strSecondMultMatr: TStringGrid;

strResultMultMatr: TStringGrid;

strPow: TStringGrid;

lbResultPow: TLabel;

strResultPow: TStringGrid;

strDet: TStringGrid;

lbResultDet: TLabel;

lbSizeAdd: TLabel;

edSize2Add: TEdit;

Label2: TLabel;

btOKSizeAdd: TButton;

edSize1Add: TEdit;

lbNumbAdd: TLabel;

edNumbAdd: TEdit;

lbSizeSub: TLabel;

Label3: TLabel;

lbNumbSub: TLabel;

edSize1Sub: TEdit;

edSize2Sub: TEdit;

btOKSizeSub: TButton;

edNumbSub: TEdit;

lbNumbMultNumb: TLabel;

Label4: TLabel;

lbSizeMultNumb: TLabel;

edSize1MultNumb: TEdit;

edSize2MultNumb: TEdit;

edNumbMultNumb: TEdit;

btOKSizeMultNumb: TButton;

Button8: TButton;

Button9: TButton;

edSize1MultMatr: TEdit;

edSize2MultMatr: TEdit;

edSize3MultMatr: TEdit;

edSize4MultMatr: TEdit;

btOKSizeMultMatr: TButton;

btOKSizePow: TButton;

edSizePow: TEdit;

edNumbPow: TEdit;

edResultDet: TEdit;

edSizeDet: TEdit;

btOKSizeDet: TButton;

tabBack: TTabSheet;

StringGrid1: TStringGrid;

StringGrid2: TStringGrid;

Edit9: TEdit;

Button17: TButton;

Button18: TButton;

GO: TButton;

GO1: TButton;

GO2: TButton;

GO3: TButton;

GO4: TButton;

GO5: TButton;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

N1: TMenuItem;

mGO: TMenuItem;

tabTranspose: TTabSheet;

strResultTranspose: TStringGrid;

strTranspose: TStringGrid;

GO6: TButton;

edSizeTranspose: TEdit;

btOKSizeTranspose: TButton;

Label1: TLabel;

btTranspose: TButton;

N2: TMenuItem;

mTranspose: TMenuItem;

OpenDialog1: TOpenDialog;

SaveDialog1: TSaveDialog;

Image1: TImage;

Memo1: TMemo;

Memo2: TMemo;

Memo3: TMemo;

Memo4: TMemo;

Memo5: TMemo;

Memo6: TMemo;

Memo7: TMemo;

Memo8: TMemo;

Memo9: TMemo;

Memo10: TMemo;

Label5: TLabel;

Memo11: TMemo;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

Label10: TLabel;

Label11: TLabel;

Memo12: TMemo;

btShowAlg: TButton;

Label12: TLabel;

Label14: TLabel;

Label15: TLabel;

Memo13: TMemo;

Button1: TButton;

Label13: TLabel;

Label16: TLabel;

Memo14: TMemo;

Image2: TImage;

Button6: TButton;

Button7: TButton;

Memo15: TMemo;

Memo16: TMemo;

Edit1: TEdit;

Label17: TLabel;

Label18: TLabel;

Label19: TLabel;

Image3: TImage;

Button10: TButton;

Button11: TButton;

MediaPlayer1: TMediaPlayer;

procedure btCalcClick(Sender: TObject);

procedure mAboutProgrammClick(Sender: TObject);

procedure btAddClick(Sender: TObject);

procedure btSubClick(Sender: TObject);

procedure btMultNumbClick(Sender: TObject);

procedure btMultMatrClick(Sender: TObject);

procedure btPowClick(Sender: TObject);

procedure btDetClick(Sender: TObject);

procedure btBackToMainClick(Sender: TObject);

procedure btAddMatrClick(Sender: TObject);

procedure btOKSizeAddClick(Sender: TObject);

procedure btAddNumbClick(Sender: TObject);

procedure btSubMatrClick(Sender: TObject);

procedure btSubNumbClick(Sender: TObject);

procedure btOKSizeSubClick(Sender: TObject);

procedure btOKSizeMultNumbClick(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button9Click(Sender: TObject);

procedure edSize1AddChange(Sender: TObject);

procedure edSize1SubChange(Sender: TObject);

procedure edSize1MultNumbChange(Sender: TObject);

procedure btOKSizeMultMatrClick(Sender: TObject);

procedure btOKSizePowClick(Sender: TObject);

procedure btOKSizeDetClick(Sender: TObject);

procedure Button17Click(Sender: TObject);

procedure GOClick(Sender: TObject);

procedure edSize1MultMatrChange(Sender: TObject);

procedure edSizePowChange(Sender: TObject);

procedure edSizeDetChange(Sender: TObject);

procedure btOKSizeTransposeClick(Sender: TObject);

procedure edSizeTransposeChange(Sender: TObject);

procedure btTransposeClick(Sender: TObject);

procedure mInstClick(Sender: TObject);

procedure mExitClick(Sender: TObject);

procedure mOpenClick(Sender: TObject);

procedure mSaveClick(Sender: TObject);

procedure mNewClick(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure btShowAlgClick(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button10Click(Sender: TObject);

procedure Button11Click(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

procedure cleanall;

var

fmOper: TfmOper;

size1, size2, size1mult, size2mult, sizeresmult1, sizeresmult2, n, numb1 : integer;

numb : real;

iden, sign, mov : byte;

f : textfile;

implementation

uses

uAboutProgramm, uCalculator, uOpenUChoise, uInstruction, uChoise, uActions, uHi, uMultMatr, uSound;

{$R \*.dfm}

const

t = 0.000001;

**///////////////////////////////////////////////////////////////////////////**

**//раздел вызова сторонних форм и реакций на кнопки на Главной странице(2)**

**//процедура активации подпунктов пункта Файл программног меню**

procedure MenuFile;

begin

with fmOper do

begin

mNew.Enabled := true;

mOpen.Enabled := true;

mSave.Enabled := true;

end;

end;

**//процедура активации вкладок формы вычислений**

procedure Click1(Page : TTabSheet);

begin

with fmOper do

begin

Page.TabVisible := true;

Page.Show;

if mov = 1 then

mOpen.Enabled := false

else

Menufile;

mov := 0;

end;

end;

**//процедура активации кнопки выполнения**

procedure Enabling(Ok : TButton);

begin

with fmOper do

begin

Ok.Enabled := false;

mGo.Enabled := false;

end;

end;

**//процедура блокировки вкладок**

procedure ShowAndExit;

begin

with fmOper do

begin

tabAdd.TabVisible := false;

tabSub.TabVisible := false;

tabMultNumb.TabVisible := false;

tabMultMatr.TabVisible := false;

tabDet.TabVisible := false;

tabPow.TabVisible := false;

tabTranspose.TabVisible := false;

end;

end;

**//процедура полной очистки вкладок формы вычислений**

procedure CleanAll;

begin

with fmOper do

begin

clean(strFirstAdd);

clean(strSecondAdd);

clean(strResultAdd);

edSize1Add.Text := '';

edSize2Add.Text := '';

edNumbAdd.Text := '';

clean(strFirstSub);

clean(strSecondSub);

clean(strResultSub);

edSize1Sub.Text := '';

edSize2Sub.Text := '';

edNumbSub.Text := '';

clean(strMultNumb);

clean(strResultMultNumb);

edSize1MultNumb.Text := '';

edSize2MultNumb.Text := '';

edNumbMultNumb.Text := '';

clean(strFirstMultMatr);

clean(strSecondMultMatr);

clean(strResultMultMatr);

edSize1MultMatr.Text := '';

edSize2MultMatr.Text := '';

edSize1MultMatr.Text := '';

edSize2MultMatr.Text := '';

clean(strPow);

clean(strResultPow);

edSizePow.Text := '';

edNumbPow.Text := '';

clean(strDet);

edResultDet.Text := '';

edSizeDet.Text := '';

clean(strResultTranspose);

clean(strTranspose);

edSizeTranspose.Text := '';

end;

end;

**//процедура открытия калькулятора и создания временного файла для истории вычислений**

procedure TfmOper.btCalcClick(Sender: TObject);

begin

fmCalculator.EPoleEnter.Text:= '';

fmCalculator.Show;

assignfile(f, 'Занесение в файл.txt');

rewrite(f);

closefile(f);

btCalc.Enabled := false;

end;

procedure TfmOper.btAddClick(Sender: TObject);

begin

mov := 1;

Click1(tabAdd);

end;

procedure TfmOper.btSubClick(Sender: TObject);

begin

mov := 1;

Click1(tabSub);

end;

procedure TfmOper.btMultNumbClick(Sender: TObject);

begin

Click1(tabMultNumb);

end;

procedure TfmOper.btMultMatrClick(Sender: TObject);

begin

Click1(tabMultMatr);

end;

procedure TfmOper.btPowClick(Sender: TObject);

begin

Click1(tabPow);

end;

procedure TfmOper.btDetClick(Sender: TObject);

begin

Click1(tabDet);

end;

procedure TfmOper.btTransposeClick(Sender: TObject);

begin

Click1(tabTranspose);

end;

procedure TfmOper.btBackToMainClick(Sender: TObject);

begin

fmOper.tabMain.Show;

end;

procedure TfmOper.FormShow(Sender: TObject);

begin

with fmOper do

begin

ShowAndExit;

tabMain.Show;

mediaplayer1.Visible := false;

end;

end;

**//конец раздела (2)**

**////////////////////////////////////////////////////////////////////////////**

**////////////////////////////////////////////////////////////////////////////**

**//раздел Сложение (3)**

procedure TfmOper.btAddMatrClick(Sender: TObject);

begin

with fmOper do

begin

lbFirstAdd.Visible := true;

lbSecondAdd.Visible := true;

strSecondAdd.Visible := true;

lbSizeAdd.Visible := true;

Label2.Visible := true;

edSize2Add.Visible := true;

edSize1Add.Visible := true;

btOKSizeAdd.Visible := true;

edNumbAdd.Visible :=false;

lbNumbAdd.Visible := false;

memo7.Visible := false;

memo6.Visible := true;

end;

MenuFile;

iden := 0;

end;

**//задание размерностей матриц и числа(если необоходимо)**

procedure TfmOper.btOKSizeAddClick(Sender: TObject);

var

flag : boolean;

begin

with fmOper do

begin

flag := true;

CheckEdit(edSize1Add, flag);

if flag = false then exit;

CheckEdit(edSize2Add, flag);

if flag = false then exit;

SetEdit(edSize1Add, size1);

SetEdit(edSize2Add, size2);

if iden <> 0 then

begin

CheckEdit(edNumbAdd, flag);

if flag = false then exit;

if edNumbAdd.Text = '' then

numb :=0

else

numb := strtofloat(edNumbAdd.Text);

edNumbAdd.Text := floattostr(numb);

end;

GO.Enabled := true;

mGO.Enabled := true;

end;

end;

procedure TfmOper.btAddNumbClick(Sender: TObject);

begin

with fmOper do

begin

edSize2Add.Visible := true;

edSize1Add.Visible := true;

edNumbAdd.Visible := true;

lbNumbAdd.Visible := true;

btOKSizeAdd.Visible := true;

lbFirstAdd.Visible := false;

lbSecondAdd.Visible := false;

strSecondAdd.Visible := false;

memo7.Visible := true;

label2.Visible := true;

lbsizeadd.Visible := true;

memo6.Visible := false;

end;

MenuFile;

iden := 1;

end;

procedure TfmOper.edSize1AddChange(Sender: TObject);

begin

Enabling(Go);

end;

**//конец раздела (3)**

**//////////////////////////////////////////////////////////////////////////////**

**//////////////////////////////////////////////////////////////////////////////**

**//радел Вычитание (4)**

procedure TfmOper.btSubMatrClick(Sender: TObject);

begin

with fmOper do

begin

lbFirstSub.Visible := true;

lbSecondSub.Visible := true;

strSecondSub.Visible := true;

lbSizeSub.Visible := true;

Label2.Visible := true;

edSize2Sub.Visible := true;

edSize1Sub.Visible := true;

btOKSizeSub.Visible := true;

edNumbSub.Visible :=false;

lbNumbSub.Visible := false;

label3.Visible := true;

memo8.Visible := true;

memo9.Visible := false;

MenuFile;

iden := 0;

end;

end;

procedure TfmOper.btSubNumbClick(Sender: TObject);

begin

with fmOper do

begin

edSize2Sub.Visible := true;

edSize1Sub.Visible := true;

edNumbSub.Visible := true;

lbNumbSub.Visible := true;

btOKSizeSub.Visible := true;

lbFirstSub.Visible := false;

lbSecondSub.Visible := false;

strSecondSub.Visible := false;

label3.Visible := true;

lbSizeSub.Visible := true;

memo8.Visible := false;

memo9.Visible := true;

MenuFile;

iden := 1;

end;

end;

**//задание размерностей матриц и числа(если необоходимо)**

procedure TfmOper.btOKSizeSubClick(Sender: TObject);

var

flag : boolean;

begin

with fmOper do

begin

flag := true;

CheckEdit(edSize1Sub, flag);

if flag = false then exit;

CheckEdit(edSize2Sub, flag);

if flag = false then exit;

SetEdit(edSize1Sub, size1);

SetEdit(edSize2Sub, size2);

if iden <> 0 then

begin

CheckEdit(edNumbSub, flag);

if flag = false then exit;

if edNumbSub.Text = '' then

numb :=0

else

numb := strtofloat(edNumbSub.Text);

edNumbSub.Text := floattostr(numb);

end;

Go1.Enabled := true;

mGO.Enabled := true;

end;

end;

**//вычитание из первого второе**

procedure TfmOper.Button8Click(Sender: TObject);

begin

with fmOper do

begin

btSubMatr.Enabled := true;

btSubNumb.Enabled := true;

RadioButton1.Checked := true;

RadioButton2.Checked := false;

sign := 1;

end;

end;

**//вычитание из второго первого**

procedure TfmOper.Button9Click(Sender: TObject);

begin

with fmOper do

begin

btSubMatr.Enabled := true;

btSubNumb.Enabled := true;

RadioButton2.Checked := true;

RadioButton1.Checked := false;

sign := 0;

end;

end;

procedure TfmOper.edSize1SubChange(Sender: TObject);

begin

Enabling(Go1);

end;

**//конец раздела (4)**

**/////////////////////////////////////////////////////////////////////////////**

**////////////////////////////////////////////////////////////////////////////**

**//раздел Умножение на число (5)**

**//задание размерностей матриц и числа**

procedure TfmOper.btOKSizeMultNumbClick(Sender: TObject);

var

flag : boolean;

begin

with fmOper do

begin

flag := true;

CheckEdit(edSize1MultNumb, flag);

if flag = false then exit;

CheckEdit(edSize2MultNumb, flag);

if flag = false then exit;

SetEdit(edSize1MultNumb, size1);

SetEdit(edSize2MultNumb, size2);

CheckEdit(edNumbMultNumb, flag);

if flag = false then exit;

if edNumbMultNumb.Text = '' then

numb :=0

else numb := strtofloat(edNumbMultNumb.Text);

edNumbMultNumb.Text := floattostr(numb);

Go2.Enabled := true;

mGO.Enabled := true;

end;

end;

procedure TfmOper.edSize1MultNumbChange(Sender: TObject);

begin

Enabling(Go2);

end;

**//конец раздела (5)**

**////////////////////////////////////////////////////////////////////////////**

**////////////////////////////////////////////////////////////////////////////**

**//раздел Перемножение матриц (6)**

**//задание размерностей матриц**

procedure TfmOper.btOKSizeMultMatrClick(Sender: TObject);

var

flag : boolean;

begin

with fmOper do

begin

flag := true;

CheckEdit(edSize1MultMatr, flag);

if flag = false then exit;

CheckEdit(edSize2MultMatr, flag);

if flag = false then exit;

CheckEdit(edSize4MultMatr, flag);

if flag = false then exit;

SetEdit(edSize1MultMatr, size1);

SetEdit(edSize2MultMatr, size2);

size1mult := size2;

edSize3MultMatr.Text := inttostr(size1mult);

SetEdit(edSize4MultMatr, size2mult);

sizeresmult1 := size1;

sizeresmult2 := size2mult;

Go3.Enabled := true;

mGO.Enabled := true;

end;

end;

procedure TfmOper.edSize1MultMatrChange(Sender: TObject);

begin

Enabling(Go3);

end;

**//конец раздела (6)**

**////////////////////////////////////////////////////////////////////////////**

**////////////////////////////////////////////////////////////////////////////**

**//раздел Возведение в степень (7)**

**//задание размерности матрицы и числа**

procedure TfmOper.btOKSizePowClick(Sender: TObject);

var

flag : boolean;

begin

with fmOper do

begin

flag := true;

CheckEdit(edSizePow, flag);

if flag = false then exit;

CheckEdit(edNumbPow, flag);

if flag = false then exit;

SetEdit(edSizePow, size1);

SetEdit(edNumbPow, numb1);

Go4.Enabled := true;

mGO.Enabled := true;

end;

end;

procedure TfmOper.edSizePowChange(Sender: TObject);

begin

Enabling(Go4);

end;

**//конец раздела (7)**

**////////////////////////////////////////////////////////////////////////////**

**////////////////////////////////////////////////////////////////////////////**

**//раздел Определитель (8)**

**////задание размерности матрицы**

procedure TfmOper.btOKSizeDetClick(Sender: TObject);

var

flag : boolean;

begin

with fmOper do

begin

flag := true;

CheckEdit(edSizeDet, flag);

if flag = false then exit;

SetEdit(edSizeDet, size1);

GO5.Enabled := true;

mGO.Enabled := true;

end;

end;

procedure TfmOper.edSizeDetChange(Sender: TObject);

begin

Enabling(Go5);

end;

**//конец раздела (8)**

**////////////////////////////////////////////////////////////////////////////**

**////////////////////////////////////////////////////////////////////////////**

**//раздел Транспонирование (10)**

**////задание размерностей матрицы**

procedure TfmOper.btOKSizeTransposeClick(Sender: TObject);

var

flag : boolean;

begin

with fmOper do

begin

flag := true;

CheckEdit(edSizeTranspose, flag);

if flag = false then exit;

CheckEdit(Edit1, flag);

if flag = false then exit;

SetEdit(edSizeTranspose, size1);

SetEdit(Edit1, size2);

GO6.Enabled := true;

mGO.Enabled := true;

end;

end;

procedure TfmOper.edSizeTransposeChange(Sender: TObject);

begin

Enabling(Go6);

end;

**//конец раздела (10)**

**////////////////////////////////////////////////////////////////////////////**

**/////////////////////////////////////////////////////////////////////////////**

**//раздел Кнопка выполнения ()**

procedure TfmOper.GOClick(Sender: TObject);

begin

with fmOper do

begin

if tabAdd.Showing = true then add;

if tabSub.Showing = true then sub;

if tabMultNumb.Showing = true then multnumb;

if tabMultMatr.Showing = true then multmatrix;

if tabPow.Showing = true then pow;

if tabDet.Showing = true then determinate;

if tabTranspose.Showing = true then transpose;

end;

end;

**//////////////////////////////////////////////////////////////////////////////**

**//раздел Пункты меню**

**//закрытие пункта Открыть в случае некорректности данных**

procedure ExitOpen;

begin

messagebox(fmOper.Handle, pchar('Некорректные данные! Результаты так же неверны!'), pchar('Ошибка'), mb\_ok+mb\_iconerror);

if (sound = true) and (player = false) then

Music('Предупреждения\Ошибки в расчетах.mp3');

Exit;

end;

**//открытие из файла**

procedure TfmOper.mOpenClick(Sender: TObject);

var

i, j : integer;

a : real;

begin

with fmOper do

begin

if tabPow.Showing = true then

begin

clean(strPow);

clean(strResultPow);

if OpenDialog1.Execute then

try

begin

assignfile(f, OpenDialog1.FileName);

reset(f);

read(f, size1);

read(f, numb1);

edSizePow.Text := inttostr(size1);

edNumbPow.text := inttostr(numb1);

for i := 0 to size1-1 do

for j := 0 to size1-1 do

begin

read(f, a);

strPow.Cells[j+1, i+1] := floattostr(a);

end;

closefile(f);

pow;

end;

GO4.Enabled := true;

except

ExitOpen;

end;

end;

if tabMultNumb.Showing = true then

begin

clean(strMultNumb);

clean(strResultMultNumb);

if OpenDialog1.Execute then

try

begin

assignfile(f, OpenDialog1.FileName);

reset(f);

read(f, size1);

read(f, size2);

read(f, numb);

edSize1MultNumb.Text := inttostr(size1);

edSize2MultNumb.Text := inttostr(size2);

edNumbMultNumb.Text := floattostr(numb);

for i := 0 to size1-1 do

for j := 0 to size1-1 do

begin

read(f, a);

strMultNumb.Cells[j+1, i+1] := floattostr(a);

end;

closefile(f);

multnumb;

end;

GO2.Enabled := true;

except

ExitOpen;

end;

end;

if tabDet.Showing = true then

begin

clean(fmoper.strDet);

if OpenDialog1.Execute then

try

begin

assignfile(f, OpenDialog1.FileName);

reset(f);

read(f, size1);

edSizeDet.Text := inttostr(size1);

for i := 0 to size1-1 do

for j := 0 to size1-1 do

begin

read(f, a);

strDet.Cells[j+1, i+1] := floattostr(a);

end;

closefile(f);

determinate;

end;

GO5.Enabled := false;

except

ExitOpen;

end;

end;

if tabTranspose.Showing = true then

begin

clean(fmOper.strTranspose);

clean(fmOper.strResultTranspose);

if OpenDialog1.Execute then

try

begin

assignfile(f, OpenDialog1.FileName);

reset(f);

read(f, size1);

edSizeTranspose.Text := inttostr(size1);

for i := 0 to size1-1 do

for j := 0 to size1-1 do

begin

read(f, a);

strTranspose.Cells[j+1, i+1] := floattostr(a);

end;

closefile(f);

transpose;

end;

GO6.Enabled := true;

except

ExitOpen;

end;

end;

if tabMultMatr.Showing = true then

begin

clean(strFirstMultMatr);

clean(strSecondMultMatr);

clean(strResultMultMatr);

if OpenDialog1.Execute then

try

begin

assignfile(f, OpenDialog1.FileName);

reset(f);

read(f, size1);

read(f, size2);

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

read(f, a);

strFirstMultMatr.Cells[j+1, i+1] := floattostr(a);

end;

//read(f, size1mult);

size1mult := size2;

read(f, size2mult);

sizeresmult1 := size1;

sizeresmult2 := size2mult;

for i := 0 to size1mult-1 do

for j := 0 to size2mult-1 do

begin

read(f, a);

strSecondMultMatr.Cells[j+1, i+1] := floattostr(a);

end;

edSize1MultMatr.Text := inttostr(size1);

edSize2MultMatr.Text := inttostr(size2);

edSize3MultMatr.Text := inttostr(size1mult);

edSize4MultMatr.Text := inttostr(size2mult);

closefile(f);

multmatrix;

end;

GO3.Enabled := true;

except

ExitOpen;

end;

end;

if tabAdd.Showing = true then

begin

clean(strResultAdd);

clean(strFirstAdd);

clean(strSecondAdd);

edSize2Add.Text := '';

edSize1Add.Text := '';

edNumbAdd.Text := '';

if OpenDialog1.Execute then

try

begin

assignfile(f, OpenDialog1.FileName);

reset(f);

read(f, size1);

read(f, size2);

if iden = 1 then

begin

read(f, numb);

edNumbAdd.Text := floattostr(numb);

end;

edSize2Add.Text := inttostr(size2);

edSize1Add.Text := inttostr(size1);

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

read(f, a);

strFirstAdd.Cells[j+1, i+1] := floattostr(a);

end;

if iden = 0 then

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

read(f, a);

strSecondAdd.Cells[j+1, i+1] := floattostr(a);

end;

closefile(f);

add;

end;

GO.Enabled := true;

except

ExitOpen;

end;

end;

if tabSub.Showing = true then

begin

clean(strResultSub);

clean(strFirstSub);

clean(strSecondSub);

edSize2Sub.Text := '';

edSize1Sub.Text := '';

edNumbSub.Text := '';

if OpenDialog1.Execute then

try

begin

assignfile(f, OpenDialog1.FileName);

reset(f);

read(f, size1);

read(f, size2);

if iden = 1 then

begin

read(f, numb);

edNumbSub.Text := floattostr(numb);

end;

edSize2Sub.Text := inttostr(size2);

edSize1Sub.Text := inttostr(size1);

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

read(f, a);

strFirstSub.Cells[j+1, i+1] := floattostr(a);

end;

if iden = 0 then

for i := 0 to size1-1 do

for j := 0 to size2-1 do

begin

read(f, a);

strSecondSub.Cells[j+1, i+1] := floattostr(a);

end;

closefile(f);

Sub;

end;

GO1.Enabled := true;

except

ExitOpen;

end;

end;

end;

end;

**//сохранение в файл**

procedure TfmOper.mSaveClick(Sender: TObject);

begin

with fmOper do

begin

if SaveDialog1.Execute then

begin

n := 1;

if tabPow.Showing = true then

SaveTable(strResultPow, 'возведения в степень');

if tabAdd.Showing = true then

SaveTable(strResultAdd, 'сложения');

if tabSub.Showing = true then

SaveTable(strResultSub, 'вычитания');

if tabMultNumb.Showing = true then

SaveTable(strResultMultNumb, 'умножения на число');

if tabMultMatr.Showing = true then

SaveTable(strResultMultMatr, 'перемножения матриц');

if tabDet.Showing = true then

SaveEdit(edResultDet);

if tabTranspose.Showing = true then

SaveTable(strResultTranspose, 'транспонирования');

n := 0;

end;

end;

end;

**//создание нового файла(очистка полей вкладки)**

procedure TfmOper.mNewClick(Sender: TObject);

begin

with fmOper do

if tabAdd.Showing = true then

begin

clean(strFirstAdd);

clean(strSecondAdd);

clean(strResultAdd);

edSize1Add.Text := '';

edSize2Add.Text := '';

edNumbAdd.Text := '';

end;

if tabSub.Showing = true then

begin

clean(strFirstSub);

clean(strSecondSub);

clean(strResultSub);

edSize1Sub.Text := '';

edSize2Sub.Text := '';

edNumbSub.Text := '';

end;

if tabMultNumb.Showing = true then

begin

clean(strMultNumb);

clean(strResultMultNumb);

edSize1MultNumb.Text := '';

edSize2MultNumb.Text := '';

edNumbMultNumb.Text := '';

end;

if tabMultMatr.Showing = true then

begin

clean(strFirstMultMatr);

clean(strSecondMultMatr);

clean(strResultMultMatr);

edSize1MultMatr.Text := '';

edSize2MultMatr.Text := '';

edSize1MultMatr.Text := '';

edSize2MultMatr.Text := '';

end;

if tabPow.Showing = true then

begin

clean(strPow);

clean(strResultPow);

edSizePow.Text := '';

edNumbPow.Text := '';

end;

if tabDet.Showing = true then

begin

clean(strDet);

edResultDet.Text := '';

edSizeDet.Text := '';

end;

if tabTranspose.Showing = true then

begin

clean(strResultTranspose);

clean(strTranspose);

edSizeTranspose.Text := '';

end;

end;

**//возов информации о прорамме**

procedure TfmOper.mAboutProgrammClick(Sender: TObject);

begin

fmAboutProgramm.Show;

end;

**//переход к спаравке**

procedure TfmOper.mInstClick(Sender: TObject);

begin

if not assigned(fmInstruction) then

fmInstruction := tfmInstruction.create(self);

fmInstruction.show;

fmInstruction.rchInstruction.Lines.LoadFromFile( way + 'Инструкция пользователя.txt');

end;

**//выход из приложения (полная очистка)**

procedure TfmOper.mExitClick(Sender: TObject);

begin

close;

cleanall;

ShowAndExit;

end;

**//показ алгоритмов**

procedure TfmOper.btShowAlgClick(Sender: TObject);

begin

if not assigned(fmAlgoritm) then

fmAlgoritm := tfmAlgoritm.create(self);

fmAlgoritm.show;

end;

procedure TfmOper.Button6Click(Sender: TObject);

begin

with fmOper do

begin

image2.Visible := true;

button7.Visible := true;

button6.Enabled := false;

end;

end;

procedure TfmOper.Button7Click(Sender: TObject);

begin

with fmOper do

begin

image2.Visible := false;

button7.Visible := false;

button6.Enabled := true;

end;

end;

procedure TfmOper.Button10Click(Sender: TObject);

begin

with fmOper do

begin

image3.Visible := true;

button11.Visible := true;

button10.Visible := false;

end;

end;

procedure TfmOper.Button11Click(Sender: TObject);

begin

with fmOper do

begin

image3.Visible := false;

button11.Visible := false;

button10.Visible := true;

end;

end;

procedure TfmOper.FormClose(Sender: TObject; var Action: TCloseAction);

begin

form6.Close;

fmChoise.Enabled := true;

fmChoise.Visible := true;

end;

**end.**

**Unit uPlayerOff**

**unit uPlayerOff;**

interface

uses

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

Dialogs, StdCtrls;

type

TForm2 = class(TForm)

Button1: TButton;

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

playeron : boolean;

implementation

uses uSound, uHi;

{$R \*.dfm}

**//процедура отключения плеера**

procedure TForm2.Button1Click(Sender: TObject);

begin

form1.MediaPlayer1.Destroy;

form2.Close;

playeron := false;

form2.Button1.Visible := false;

form2.Button1.Enabled := false;

end;

**end.**

**Unit uSound**

**unit uSound;**

interface

uses

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

Dialogs, StdCtrls, MPlayer;

type

TForm1 = class(TForm)

Label1: TLabel;

Button1: TButton;

Button2: TButton;

Label2: TLabel;

Button3: TButton;

Button4: TButton;

MediaPlayer1: TMediaPlayer;

procedure Button2Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure MediaPlayer1Notify(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

**//тип запись для плейлиста**

type

pt = ^TPlaylist;

TPlayList = record

name : string;

next, prev : pt;

end;

var

Form1: TForm1;

tracklist : pt;

player, sound : boolean;

MediaPlayer1 : TMediaPlayer;

implementation

uses uChoise, uHi, uPlayerOff;

{$R \*.dfm}

**//процедура создания списка (плейлиста)**

procedure CreateList;

var

i : integer;

first, last : pt;

begin

i := 0;

assignfile(f, way + 'Треки.txt');

reset(f);

New(tracklist);

while not EoF(f) do

begin

inc(i);

new(tracklist^.next);

tracklist^.next^.prev := tracklist;

tracklist := tracklist^.next;

readln(f, tracklist^.name);

if i = 1 then first := tracklist;

end;

last := tracklist;

tracklist^.next := first;

tracklist := first;

tracklist^.prev := last;

closefile(f);

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

label1.Visible := false;

label2.Visible := true;

label2.Enabled := true;

button1.Enabled := false;

button1.Visible := false;

button2.Enabled := false;

button2.Visible := false;

button3.Enabled := true;

button3.Visible := true;

button4.Enabled := true;

button4.Visible := true;

end;

**//активация плеера**

procedure TForm1.Button1Click(Sender: TObject);

begin

fmChoise.Show;

if not assigned(form2) then

form2 := tform2.create(self);

form2.Show;

form1.Visible := false;

form1.Enabled := false;

createlist;

mediaplayer1.FileName := (way + 'Треки\Матрица\' + tracklist^.name);

tracklist := tracklist^.next;

mediaplayer1.Open;

mediaplayer1.Play;

player := true;

end;

**//обращение к плееру для переключения**

procedure TForm1.MediaPlayer1Notify(Sender: TObject);

var

i : integer;

begin

mediaplayer1.FileName := (way + 'Треки\Матрица' + tracklist^.name);

tracklist := tracklist^.next;

mediaplayer1.Open;

mediaplayer1.Play;

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

sound := true;

fmChoise.Show;

form1.Visible := false;

form1.Enabled := false;

end;

procedure TForm1.Button4Click(Sender: TObject);

begin

sound := false;

fmChoise.Show;

form1.Visible := false;

form1.Enabled := false;

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);

begin

fmHi.Enabled := true;

fmHi.Visible := true;

end;

**end.**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 017ПЗ | | | | Пояснительная записка | | | | 86 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 651005  017 ПД | | | | Схема программы | | | | Формат А2 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 017 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | ПС «Матричный калькулятор»  Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Плиска В.С. |  | 25.05.17 | Т |  | |  | 86 | 86 |
| Пров. | | Болтак С.В. |  | 06.06.17 | Кафедра ПОИТ  Гр. 651005 | | | | | |
|  | |  |  |  |