**СОДЕРЖАНИЕ**

[1 Анализ литературных источников и формирование функциональных требований к разрабатываемому программному средству 2](#_Toc132918575)

[1.1 Обзор существующих аналогов 2](#_Toc132918576)

[1.1.1 Программное приложение Wolfram Mathematica 2](#_Toc132918577)

[1.1.2 Программное приложение Mathcad 14 4](#_Toc132918578)

[1.1.3 Онлайн-сервис Matrix Calculator 5](#_Toc132918579)

[1.2 Описание средств разработки 7](#_Toc132918580)

[1.2.1 Линейный однонаправленный список 7](#_Toc132918581)

[1.2.2 Стек 8](#_Toc132918582)

[1.2.3 Файлы 9](#_Toc132918583)

[1.2.4 Классы 11](#_Toc132918584)

[1.3 Спецификация функциональных требований 11](#_Toc132918585)

[2 Проектирование и разработка программного средства 15](#_Toc132918586)

[2.1 Описание алгоритмов решения задачи 15](#_Toc132918587)

[2.2 Структура данных 18](#_Toc132918588)

[2.2.1 Структура типов основной программы 18](#_Toc132918589)

[2.2.2 Структура данных основной программы 19](#_Toc132918590)

[2.2.3 Структура данных алгоритма ExprCalculation(Expression, Res) 19](#_Toc132918591)

[2.2.4 Структура данных алгоритма OperandProcessing (PointerPos, Expression, OperandStack) 20](#_Toc132918592)

[2.2.5 Структура данных алгоритма OperatorProcessing (NewOperator, OperandStack, OperatorStack) 20](#_Toc132918593)

[2.2.6 Структура типов алгоритма GetPriority (Oper, Res) 21](#_Toc132918594)

[2.2.7 Структура данных алгоритма GetPriority (Oper, Res) 21](#_Toc132918595)

[2.2.8 Структура данных алгоритма NumbToMatr (Numb, Res) 21](#_Toc132918596)

[2.2.9 Структура данных алгоритма SumMatrixes (FirstMatr, SecondMatr, Res) 22](#_Toc132918597)

[2.2.10 Структура данных алгоритма SubMatrixes (FirstMatr, SecondMatr, Res) 22](#_Toc132918598)

[2.2.11 Структура данных алгоритма MultMatrixes (FirstMatr, SecondMatr, Res) 23](#_Toc132918599)

[2.2.12 Структура данных алгоритма MultConstMatrix (Numb, FirstMatr, Res) 23](#_Toc132918600)

# Анализ литературных источников и формирование функциональных требований к разрабатываемому программному средству

## Обзор существующих аналогов

Прежде чем разрабатывать собственный продукт для выполнения операций над матрицами, следует рассмотреть существующие аналоги, проанализировать их преимущества и недостатки для улучшения качества разрабатываемого средства. Также этот анализ может дать представление о различных подходах к решению задач, связанных с матрицами, и помочь определить, какой подход наиболее подходит для конкретной ситуации.

Существует много аналогов для работы с матрицами в связи с тем, что матрицы широко используются в различных областях, таких как математика, физика, инженерия, компьютерные науки, экономика и другие. Каждая из этих областей имеет свои специфические задачи и требования к работе с матрицами, что приводит к появлению новых аналогов и инструментов.

Для анализа были выбраны следующие средства:

* Wolfram Mathematica;
* Mathcad;
* Matrix Calculator.

### Программное приложение Wolfram Mathematica

Wolfram Mathematica является мощным инструментом для работы с матрицами и линейной алгеброй. Он поддерживает множество операций над матрицами, включая:

* создание матриц;
* сложение и вычитание матриц;
* умножение матрицы на скаляр;
* умножение матрицы на матрицу;
* транспонирование матриц;
* вычисление определителя матрицы;
* нахождение обратной матрицы;
* нахождение ранга матрицы;
* нахождение собственных значений и собственных векторов матрицы;
* LU-разложение, QR-разложение и сингулярное разложение матрицы;
* решение систем линейных уравнений;
* приведение к треугольному и диагональному виду;
* вычисление скалярного и крестового произведений векторов;
* изменение размерности матрицы;
* конкатенация матриц;
* извлечение подматрицы;
* нахождение элементов матрицы по заданным условиям;
* элементарные преобразования над матрицами;
* нахождение обобщенной обратной матрицы;
* нахождения характеристического полинома матрицы;
* нахождения следа матрицы.

Некоторые из операций представлены на рисунке 1.1.

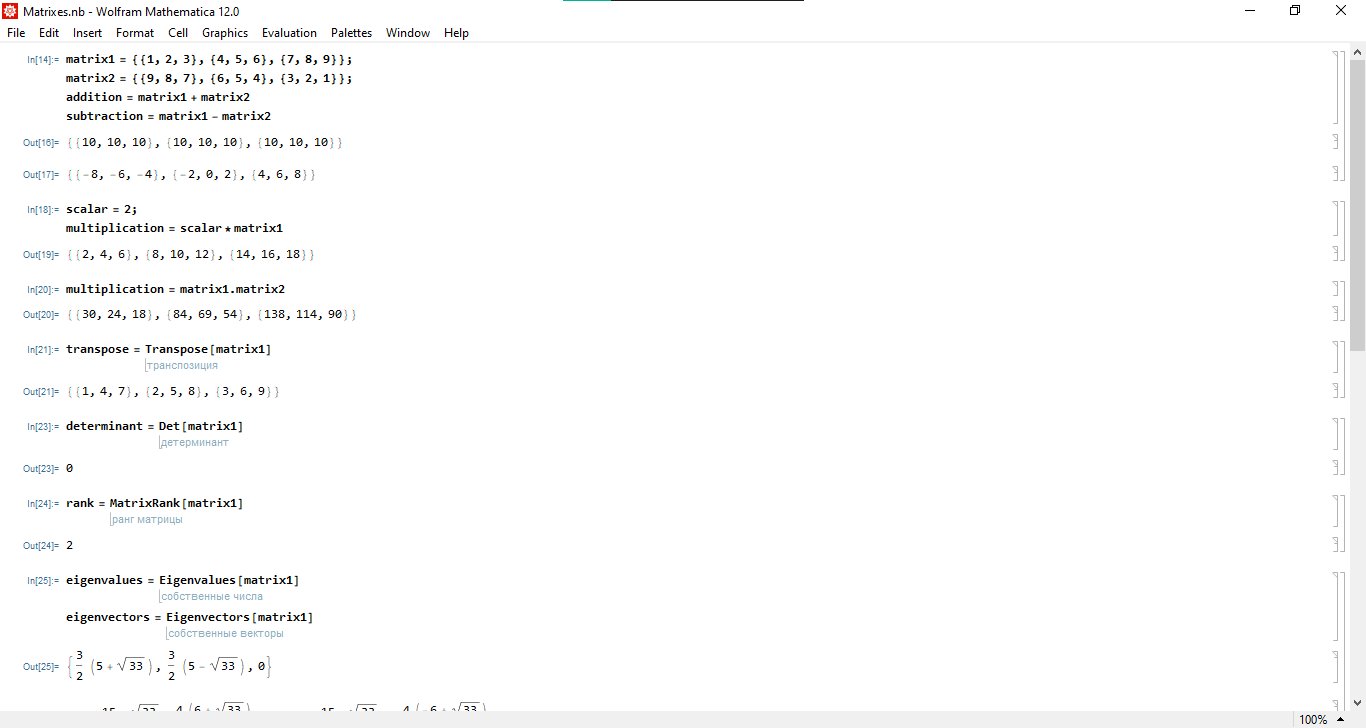


Рисунок . – Программное средство Wolfram Mathematica

Это далеко не все возможности, которые предоставляет Wolfram Mathematica для работы с матрицами. Он также поддерживает многие другие операции, такие как нахождение норм матриц, работу с разреженными матрицами и многое другое.

Преимущества функционала операций над матрицами в Wolfram Mathematica:

* широкий набор операций;
* высокая точность вычислений;
* интеграция с другими функциями;
* графический интерфейс.

Mathematica имеет широкий набор операций над матрицами и линейной алгеброй, использует символьные вычисления и может обрабатывать матрицы с высокой точностью, может интегрироваться с другими функциями и инструментами, такими как графики и анализ данных и, к тому же имеет графический интерфейс, который делает работу с матрицами более удобной и понятной для пользователей.

Недостатки функционала операций над матрицами в Wolfram Mathematica:

* высокий порог входа;
* скорость работы в сравнении с аналогичными средствами.

Mathematica может потребовать определенных знаний и навыков, чтобы использовать его на полную мощность, а также некоторые операции над матрицами в Mathematica могут быть менее эффективными, чем эквивалентные операции в специализированных библиотеках или других математических программных пакетах.

Таким образом, Wolfram Mathematica является мощным инструментом для работы с матрицами, но его сложность использования могут быть препятствием для многих пользователей. Однако, если у вас есть опыт работы с этим инструментом и у вас есть задачи, которые требуют широкого функционала, то Wolfram Mathematica будет лучшим выбором для вас.

### Программное приложение Mathcad 14

Mathcad является еще одним инструментом для работы с матрицами и линейной алгеброй. Он также поддерживает множество операций над матрицами, включая:

* создание матриц;
* сложение и вычитание матриц;
* умножение матрицы на скаляр;
* умножение матрицы на матрицу;
* транспонирование матриц;
* вычисление определителя матрицы;
* нахождение обратной матрицы;
* нахождение ранга матрицы;
* нахождение собственных значений и собственных векторов матрицы;
* LU-разложение, QR-разложение и сингулярное разложение матрицы;
* решение систем линейных уравнений;
* вычисление скалярного и крестового произведений векторов.

Это не полный список операций, но они являются основными возможностями Mathcad для работы с матрицами.

Преимущества функционала операций над матрицами в Mathcad:

* удобство в использовании;
* высокая точность вычислений;
* интеграция с другими функциями.

Некоторые из операций представлены на рисунке 1.2.

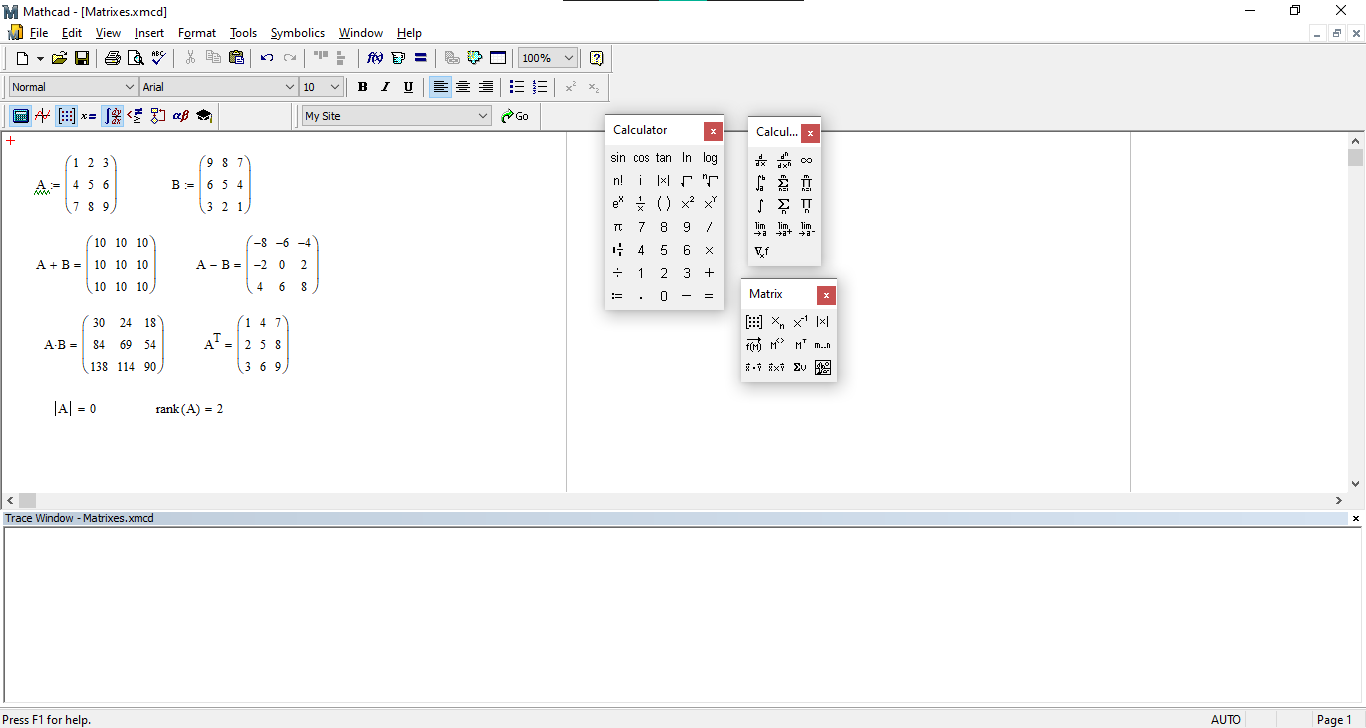


Рисунок . – Программное средство Mathcad

Mathcad имеет удобный интерфейс и может быть легко использован для выполнения операций с матрицами, к тому же он использует символьные вычисления и может обрабатывать матрицы с высокой точностью и может интегрироваться с другими функциями и инструментами, что делает его удобным инструментом для решения сложных задач.

Недостатки функционала операций над матрицами в Mathcad:

* недостаток операций;
* ограниченный выбор форматов файлов.

В Mathcad иногда для выполнения более сложных операций над матрицами может потребоваться написать сложный код, и он имеет ограниченный выбор форматов файлов для экспорта или импорта матриц и других данных.

Таким образом, Mathcad является удобным инструментом при работе

над матрицами, однако при надобности использования более сложных операций, придется написать код, что для пользователя, не знакомым с программированием, может оказаться непосильной задачей.

### Онлайн-сервис Matrix Calculator

Matrix Calculator является онлайн-сервисом, предназначенным для выполнения матричных операций. Среди возможных операций над матрицами, которые он поддерживает, можно выделить следующие:

* создание матриц;
* сложение и вычитание матриц;
* умножение матрицы на скаляр;
* умножение матрицы на матрицу;
* транспонирование матриц;
* вычисление определителя матрицы;
* нахождение обратной матрицы;
* нахождение псевдообратной матрицы;
* нахождение ранга матрицы;
* нахождение собственных значений и собственных векторов матрицы;
* LU-разложение, QR-разложение, сингулярное разложение матрицы, разложение Холецкого и разложение Жордана;
* решение систем линейных уравнений;
* приведение к треугольному и диагональному виду.

Преимущества функционала операций над матрицами в Online Matrix Calculator:

* доступность;
* простота использования.

Matrix Calculator доступен в интернете, и его можно использовать с любого устройства с доступом в Интернет, а интерфейс онлайн-калькулятора прост в использовании, что делает его доступным для широкого круга пользователей.

Недостатки функционала операций над матрицами в Online Matrix Calculator:

* ограниченный функционал;
* не поддерживает большие объемы данных;
* необходимость подключения к Интернету.

Matrix Calculator имеет ограниченный набор операций и не может обрабатывать большие объемы данных, что ограничивает его применение для решения сложных математических задач, кроме того, он требует подключения к Интернету, что может стать проблемой в случае отсутствия связи или медленного интернет-соединения.

Таким образом, Matrix Calculator является простым и доступным инструментом для выполнения базовых матричных операций, но имеет ограниченный функционал и не может быть использован для обработки больших объемов данных. Данное средство отлично подойдет простому пользователю, который далек от программирования и которому достаточно приведенного функционала.

Графический интерфейс онлайн-сервиса представлен на рисунке 1.3.

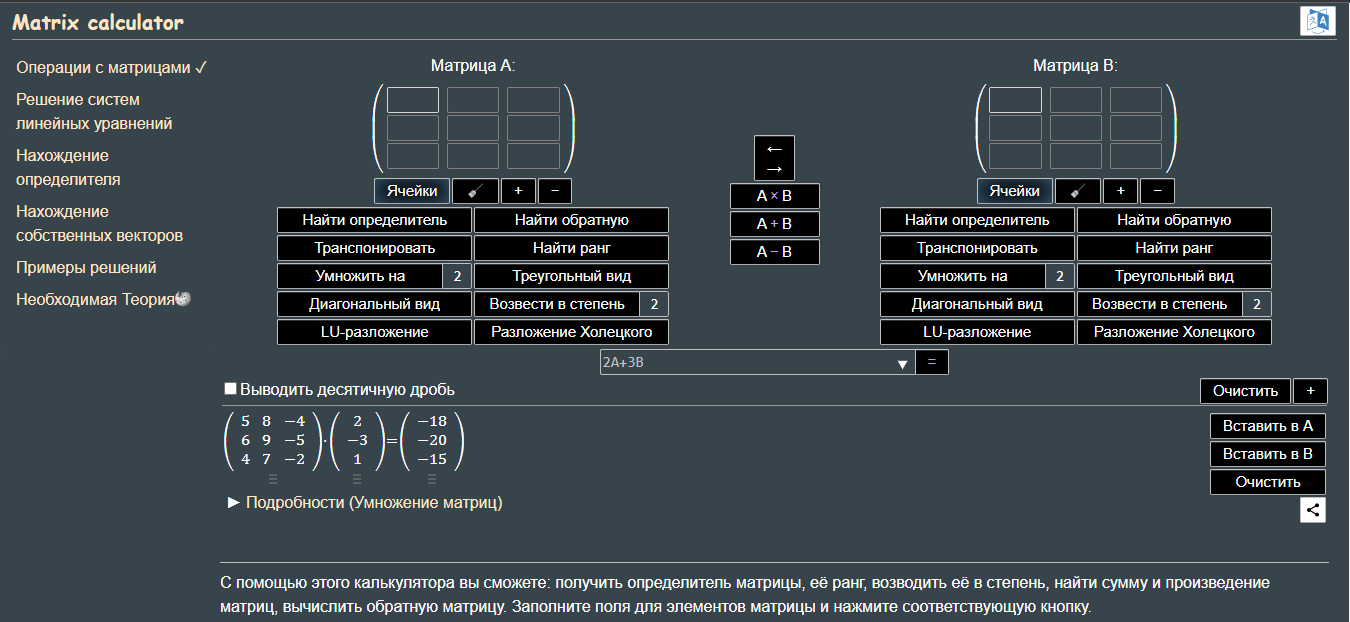


Рисунок . – Онлайн-сервис Matrix Calculator

## Описание средств разработки

Для реализации программного средства будут использоваться следующие структуры данных:

* линейный однонаправленный список;
* стек;
* файлы;
* классы.

### Линейный однонаправленный список

Линейный однонаправленный список (также называемый односвязным списком) – это структура данных, представляющая собой последовательность элементов, каждый из которых содержит какое-то значение (например, число, строку или другой объект) и ссылку на следующий элемент списка. При этом первый элемент списка имеет ссылку на второй элемент, второй элемент имеет ссылку на третий элемент, и так далее, пока не будет достигнут последний элемент, у которого ссылка на следующий элемент равна nil.

Таким образом, элементы списка связаны между собой только однонаправленной связью, что означает, что каждый их них знает только о следующем элементе, но не о предыдущем. Для доступа к элементам списка необходимо последовательно пройти по всем элементам, начиная с первого и переходя к следующему с помощью ссылки.

Структура линейного списка представлена на рисунке 1.4.

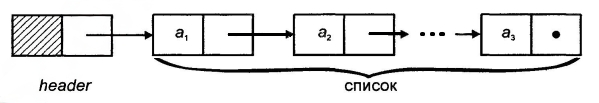


Рисунок . – Структура линейного однонаправленного списка

В данном программном средстве линейные однонаправленные списки будет целесообразным использовать для хранения матриц, а также для хранения истории выражений, введенных ранее пользователем.

### Стек

Стек – это структура данных, представляющая собой упорядоченную последовательность элементов, где каждый элемент имеет свою позицию в последовательности и доступ к элементам осуществляется только с одного конца – вершины стека.

Стек можно представить как вертикальную последовательность элементов, где вершина стека находится сверху, а основание – снизу. Элементы могут быть добавлены в стек только на вершину и удалены только с нее. Таким образом, стек работает по принципу «последним вошел – первым вышел» (LIFO – Last-In-First-Out).

Структура стека представлена на рисунке 1.5.

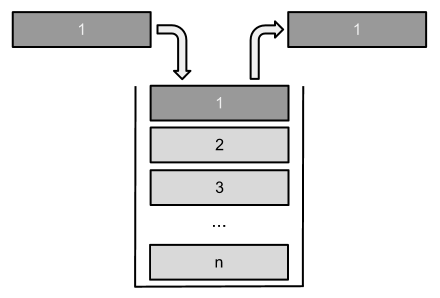


Рисунок . – Структура стека

В Delphi встроенный стек отсутствует, поэтому необходимо разработать его самостоятельно, используя либо одномерный динамический массив, либо линейный однонаправленный список.

В целом, реализация стека на массиве может быть предпочтительнее в использовании, если изначально известен максимальный размер стека и операции перевыделения памяти не происходят часто. Однако, реализация стека на линейном однонаправленном списке является более гибкой, так как не требует заранее выделения памяти, а также может быть эффективнее в случаях, когда требуется частое добавление и удаление элементов в начале списка.

В случае разрабатываемого программного средства при подсчете выражения будет использовано два стека, один из которых будет хранить операнды, а второй – операции.

В данном программном средстве будет целесообразным использование стека, реализованного на линейном однонаправленном списке, ввиду его гибкости.

### Файлы

#### 1.2.3.1 Логический и физический файл

Файлы подразделяются на две области:

* логический файл;
* физический файл.

Логический файл – это файл, который представляет собой логический набор данных, обычно имеющих определенную структуру и связанных между собой. Логический файл может использоваться для хранения различных типов данных, таких как текстовые документы, изображения, аудио- и видеофайлы, базы данных и т. д. Он определяет формат, структуру и ограничения для данных, которые он содержит.

Физический файл – а это непосредственно файл, хранящийся на жестком диске или другом устройстве хранения информации. Физический файл содержит набор битов, которые могут быть интерпретированы как данные, которые он представляет.

Логический файл и физический файл тесно связаны между собой. Логический файл описывает формат и структуру данных, которые он содержит, а физический файл представляет собой непосредственное место хранения этих данных на устройстве. Когда данные записываются в логический файл, они сохраняются в соответствующем физическом файле, который затем может быть прочитан для получения этих данных. При чтении данных из логического файла, система оперирует на физическом файле, считывая данные из определенного участка жесткого диска, и затем интерпретирует их в соответствии с форматом логического файла.

#### 1.2.3.2 Типы файлов

Всего существует три типа файлов:

* типизированные файлы;
* текстовые файлы;
* не типизированные файлы.

Типизированные файлы связываются с файловыми переменными, объявленными как file of <Тип>. Файл считается состоящим из элементов, каждый из которых имеет тип <Тип>.

Не типизированные файлы могут быть связаны только с теми файловыми переменными, которые были объявлены как file. Файл считается состоящим из элементов, размер которых определяется при открытии файла.

Текстовый файл представляет собой последовательность символов, однако, он не эквивалентен файлу типа file of Char. Файл типа file of Char – это единая последовательность символов. Текстовые файлы связываются с файловыми переменными, принадлежащими стандартному типу TextFile. Особенность текстовых файлов состоит в том, что содержащиеся в них символы разбиваются на строки. Строки могут быть различной длины (пустые в том числе). В конце каждой строки помещается специальный управляющий символ: возврат каретки (#13 или М международное обозначение CR) и перехода новую строку (#10 или международное обозначение LF). С наличием этого маркера связана логическая функция Eoln (End of line).

Структура файла представлена на рисунке 1.6.

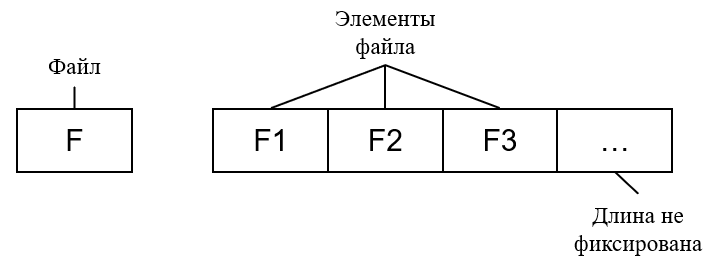


Рисунок . – Структура файла

Для сохранения файлов целесообразней использовать типизированный файл. Так как в типизированный файл можно записать данные только одного типа, разумней будет сделать его с типом record, поскольку в поля record можно будет записать данные различных типов. В поля будет помещаться основная информация, необходимая для повторного открытия файла. В данном программном средстве в поля такого файла будет помещаться:

* линейный однонаправленный список с матрицами, введенными пользователем;
* линейный однонаправленный список с историей вычисленных выражений;
* строка с выражением, введенным в поле ввода на момент сохранения;
* время открытия файла;
* время сохранения файла;
* время сессии с открытия файла до его сохранения;
* общее время редактирования файла с его создания, до его текущего сохранения;
* имя пользователя windows, создавшего файл;
* имя пользователя windows, сохранившего файл последним;
* тема оформления.

### Классы

Классы представляют собой основу объектно-ориентированного программирования (ООП), которая позволяет создавать модели объектов, объединяя вместе данные и функции, которые работают с этими данными. Классы являются абстрактными типами данных, которые содержат переменные данных и методы, определяющие поведение объектов.

Каждый класс имеет определенные свойства, методы и события, которые могут быть использованы для создания объектов класса. Свойства класса представляют данные, которые хранятся в объектах класса, а методы класса определяют функции, которые могут выполняться над объектами класса.

Использование классов позволяет группировать данные и методы, связанные с определенными структурами данных, в единый объект. Это упрощает кодирование, тестирование и поддержку программного обеспечения. В связи с этим в данном программном средстве рационально будет использовать классы для:

* реализации стека;
* реализации односвязного списка;
* матриц.

## Спецификация функциональных требований

Требование – описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной для пользователя задачи.

Функциональные требования описывают задачи, которые пользователь может выполнить, используя разрабатываемое ПС: реакцию системы на действия пользователя, сценарии работы пользователя. Функциональные требования можно также описать как функции, которые пользователь может выполнить. Важно заметить, что они отличаются от других видов требований, которые не затрагивают корректность работы программы.

Функциональные требования к разрабатываемому ПС приведены в таблице 1.

Таблица – Функциональные требования к программному средству

|  |  |
| --- | --- |
| Идентификатор | Требование |
| ФТ-1 | Создать новую матрицу с пустыми ячейками |
| ФТ-2 | Заполнить матрицу вещественными числами |
| ФТ-3 | Изменить элементы существующей матрицы |

Продолжение таблицы 1

|  |  |
| --- | --- |
| ФТ-4 | Удалить существующую матрицу |
| ФТ-5 | Ввести выражение в поле ввода с клавиатуры |
| ФТ-6 | Ввести выражение в поле ввода при помощи кнопок |
| ФТ-7 | Реализовать просмотр созданных пользователем матриц |
| ФТ-8 | Реализовать просмотр предыдущих выражений и их результатов |
| ФТ-9 | Переместить предыдущие выражение в поле ввода |
| ФТ-10 | Создать новый файл |
| ФТ-11 | Сохранить текущий файл |
| ФТ-12 | Сохранить в конкретную папку текущий файл |
| ФТ-13 | Открыть сохраненный ранее файл |
| ФТ-14 | Открыть окно с информацией о текущем файле |
| ФТ-15 | Изменить тему оформления |
| ФТ-16 | Сортировать выбранную матрицу по:   * столбцам; * строкам; * порядку |
| ФТ-17 | Сортировать список матриц по именам |

ФТ-1 Создать новую матрицу с пустыми ячейками – Пользователь должен иметь возможность создавать новые матрицы, выбирая количество строк и столбцов. При создании новой матрицы все ее ячейки должны быть пустыми.

ФТ-2 Заполнить матрицу вещественными числами – Пользователь должен иметь возможность заполнять ячейки матрицы вещественными числами.

ФТ-3 Изменить элементы существующей матрицы – Пользователь должен иметь возможность изменять значения элементов матрицы в любое время. Для этого пользователь должен выбрать ячейку, которую он хочет изменить, и ввести новое значение.

ФТ-4 Удалить существующую матрицу – Пользователь должен иметь возможность удалить существующую матрицу. При удалении матрицы должно быть предупреждение о том, что все данные будут утеряны, и пользователь должен подтвердить свой выбор.

ФТ-5 Ввести выражение в поле ввода с клавиатуры – Пользователь должен иметь возможность вводить математические выражения в поле ввода с клавиатуры.

ФТ-6 Ввести выражение в поле ввода при помощи кнопок – Пользователь должен иметь возможность вводить математические выражения в поле ввода при помощи кнопок. Для этого должны быть предоставлены кнопки со всеми математическими операторами и числами, а также с латинской раскладкой клавиатуры.

ФТ-7 Реализовать просмотр созданных пользователем матриц – Пользователь должен иметь возможность просмотреть все созданные им матрицы. Для этого должна быть предоставлена возможность просмотра списка матриц и выбора нужной матрицы.

ФТ-8 Реализовать просмотр предыдущих выражений и их результатов – программное средство должно иметь функционал, позволяющий пользователю просмотреть предыдущие введенные им выражения и их результаты. Для этого необходимо создать отдельное окно, в котором будут отображаться все предыдущие выражения и результаты их выполнения.

ФТ-9 Переместить предыдущие выражение в поле ввода – пользователь должен иметь возможность выбрать одно из предыдущих выражений и при помощи двойного щелчка левой кнопки мыши переместить его в поле ввода для дальнейшей работы с ним.

ФТ-10 Создать новый файл – требуется реализовать функционал, позволяющий пользователю создавать новые файлы для хранения матриц и других данных. Для этого необходимо создать интерфейс, который будет позволять пользователю задавать имя файла и его расположение.

ФТ-11 Сохранить текущий файл – пользователь должен иметь возможность сохранить текущий файл с матрицами и другими данными. Для этого необходимо реализовать функцию, которая будет сохранять текущий файл в его текущем расположении.

ФТ-12 Сохранить в конкретную папку текущий файл – пользователь должен иметь возможность сохранить текущий файл в заданной папке. Для этого необходимо реализовать функцию, которая будет позволять пользователю выбрать нужную папку и сохранить файл в ней.

ФТ-13 Открыть сохраненный ранее файл - пользователь должен иметь возможность открыть сохраненный ранее файл с матрицами и другими данными. Для этого необходимо реализовать функцию, которая будет позволять пользователю открывать сохраненный ранее файл.

ФТ-14 Открыть окно с информацией о текущем файле - Пользователь должен иметь возможность открыть окно, в котором будет отображена информация о текущем открытом файле. Это должно включать в себя:

* время открытия файла;
* время сохранения файла;
* время сессии с открытия файла до его сохранения;
* общее время редактирования файла с его создания, до его текущего сохранения;
* имя пользователя windows, создавшего файл;
* имя пользователя windows, сохранившего файл последним.

ФТ-15 Изменить тему оформления - Пользователь должен иметь возможность изменить тему оформления программного средства. Это может включать в себя изменение цветовой схемы, шрифта и других элементов интерфейса.

ФТ-16 Сортировать выбранную матрицу по столбцам - Пользователь должен иметь возможность отсортировать выбранную матрицу по значениям столбцов в порядке возрастания или убывания.

Сортировать выбранную матрицу по строкам - Пользователь должен иметь возможность отсортировать выбранную матрицу по значениям строк в порядке возрастания или убывания.

Сортировать список матриц по именам по порядку – Пользователь должен иметь возможность отсортировать матрицу в порядке чтения текста, а именно слева направо, сверху вниз.

ФТ-17 Сортировать список матриц по именам - Пользователь должен иметь возможность отсортировать список всех созданных матриц по их именам в алфавитном порядке.

# Проектирование и разработка программного средства

## Описание алгоритмов решения задачи

Таблица – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Описание алгоритма | Формальные  параметры | Предполагаемый тип реализации |
| 1 | Инициализация формы | Открывает форму Main. Использует следующие алгоритмы: ExprCalculation |  |  |
| 2 | Expr  Calculation  (  Expression  ) | Идет вдоль передаваемого выражения и определяет, чем является следующий символ: началом операнда или оператором. Вызывает подпрограммы OperandProcessing, OperatorProcessing | Expression – получает от фактического параметра адрес c защитой;  Res – получает от фактического параметра адрес, возвращаемый параметр | Функция. Res – возвращаемый функцией параметр |
| 3 | Operand  Processing  (  PointerPos,  Expression, OperandStack  ) | Считывает операнд в стек операндов с указателя выражения PointerPos по следующий оператор | PointerPos – получает от фактического параметра адрес;  Expression – получает от фактического параметра адрес c защитой;  OperandStack – получает от фактического параметра адрес | Процедура |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | Operator  Processing  (  NewOperator, OperandStack,  OperatorStack  ) | В зависимости от приоритета переданного оператора выполняет расчет предыдущих операций, преобразует операнды к матричному типу и вычисляет соответствующие выражения. Вызывает подпрограммы  GetPriority,  NumbToMatr,  SumMatrixes,  SubMatrixes,  MultConstMatrix,  MultMatrixes, | NewOperator – получает от фактического параметра адрес с защитой;  OperandStack – получает от фактического параметра адрес;  OperatorStack – получает от фактического параметра адрес | Процедура |
| 5 | Get  Priority  (  Oper,  Res  ) | В зависимости от переданного оператора возвращает его приоритет | Oper – получает от фактического параметра адрес с защитой;  Res – получает от фактического параметра адрес, возвращаемый параметр | Функция. Res – возвращаемый функцией параметр |
| 6 | NumbToMatr  (  Numb,  Res  ) | Преобразует число в матрицу 1x1 | Numb – получает от фактического параметра адрес с защитой;  Res – получает от фактического параметра адрес, возвращаемый параметр | Функция. Res – возвращаемый функцией параметр |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | SumMatrixes  (  FirstMatr,  SecondMatr,  Res  ) | Рассчитывает сумму переданных матриц | FirstMatr – получает от фактического параметра адрес с защитой;  SecondMatr – получает от фактического параметра адрес с защитой;  Res – получает от фактического параметра адрес, возвращаемый параметр | Функция. Res – возвращаемый функцией параметр |
| 8 | SubMatrixes  (  FirstMatr,  SecondMatr,  Res  ) | Рассчитывает разность переданных матриц | FirstMatr – получает от фактического параметра адрес с защитой;  SecondMatr – получает от фактического параметра адрес с защитой;  Res – получает от фактического параметра адрес, возвращаемый параметр | Функция. Res – возвращаемый функцией параметр |
| 9 | MultConst  Matrix  (  Numb,  FirstMatr,  Res  ) | Рассчитывает произведение числа на матрицу | Numb – получает от фактического параметра адрес с защитой;  FirstMatr – получает от фактического параметра адрес с защитой;  Res – получает от фактического параметра адрес, возвращаемый параметр | Функция. Res – возвращаемый функцией параметр |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | MultMatrixes  (  FirstMatr,  SecondMatr,  Res  ) | Рассчитывает произведение переданных матриц | FirstMatr – получает от фактического параметра адрес с защитой;  SecondMatr – получает от фактического параметра адрес с защитой;  Res – получает от фактического параметра адрес, возвращаемый параметр | Функция. Res – возвращаемый функцией параметр |

## Структура данных

### Структура типов основной программы

Таблица – Структура типов основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TMatrix | Record  FLinesAmount: Integer;  FColumnsAmount: Integer;  FElements: array of array of Real;  End | Тип, предназначенный для обозначения матриц:  FLinesAmount – количество строк;  FColumnsAmount – количество столбцов;  FElements – элементы матрицы |
| TStack<T> | Record  TStackSegment: Record;  PStackSegment: ^TStackSegment;  FTop: PStackSegment;  FSize: Integer;  End | Тип, предназначенный для обозначения стека.  TStackSegment – сегмент стека;  PStackSegment – указатель на сегмент стека;  FTop – указатель на верхний элемент стека;  FSize – количество элементов в стеке |
| TStackSegment | Record  Value: T;  Next: PStackSegment;  End | Тип, предназначенный для обозначения сегмента стека.  Value – значение типа, заданного пользователем;  Next – указатель на следующий сегмент стека |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| TSingleLinkedList<T> | Record  TNode: Record;  PNode: ^TNode;  FHead: PNode;  FSize: Integer;  End | Тип, предназначенный для обозначения однонаправленного списка.  TNode – сегмент списка;  PNode – указатель на сегмент списка;  FHead – указатель на первый элемент списка;  FSize – количество элементов в списке |
| TNode | Record  Value: T;  Next: PNode;  End | Тип, предназначенный для обозначения сегмента стека.  Value – значение типа, заданного пользователем;  Next – указатель на следующий сегмент стека |

### Структура данных основной программы

Таблица – Структура данных основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| MatrixList | TSingleLinkedList<TMatrix> | Содержит список введенных матриц |
| HistryList | TSingleLinkedList<String> | Содержит список предыдущих рассчитанных выражений |
| CursPos | Integer | Вспомогательная переменная для определения, для определения позиции курсора в строке ввода |

### Структура данных алгоритма ExprCalculation(Expression, Res)

Таблица – Структура данных алгоритма ExprCalculation(Expression, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i | Integer | Счетчик циклов | Локальный |
| Expression | String | Выражение, введенное пользователем | Формальный |

Продолжение таблицы 5

|  |  |  |  |
| --- | --- | --- | --- |
| Res | TMatrix | Возвращаемый результат выражения | Формальный |

### Структура данных алгоритма OperandProcessing (PointerPos, Expression, OperandStack)

Таблица 6 – Структура данных алгоритма OperandProcessing (PointerPos, Expression, OperandStack)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i | Integer | Счетчик циклов | Локальный |
| Operand | String | Строка накапливания текущего операнда | Локальный |
| PointerPos | Integer | Указатель на начало текущего операнда | Формальный |
| Expression | String | Выражение, введенное пользователем | Формальный |
| OperandStack | TStack<String> | Стек, содержащий операнды | Формальный |

### Структура данных алгоритма OperatorProcessing (NewOperator, OperandStack, OperatorStack)

Таблица 7 – Структура данных алгоритма OperatorProcessing (NewOperator, OperandStack, OperatorStack)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| FirstOperand | Double | Первый операнд текущей операции | Локальный |
| Second  Operand | Double | Второй операнд текущей операции | Локальный |
| NewOperator | Char | Текущий оператор в выражении | Формальный |
| OperandStack | TStack<String> | Стек, содержащий операнды | Формальный |
| OperatorStack | TStack<Char> | Стек, содержащий операторы | Формальный |

### Структура типов алгоритма GetPriority (Oper, Res)

Таблица – Структура типов алгоритма GetPriority (Oper, Res)

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| OperatorsPriority | Record  Oper: char;  Priority: byte;  End; | Тип, предназначенный для хранения приоритета оператора:  Oper – оператор;  Priority – приоритет оператора |

### Структура данных алгоритма GetPriority (Oper, Res)

Таблица – Структура данных алгоритма GetPriority (Oper, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Operators | Array [1..6] of OperatorsPriority = (  (Oper: '+'; Priority: 1),  (Oper: '-'; Priority: 1),  (Oper: '\*'; Priority: 2),  (Oper: '/'; Priority: 2),  (Oper: '('; Priority: 0),  (Oper: ')'; Priority: 255)  ); | Массив приоритетов операторов | Локальный |
| Oper | Char | Текущий оператор | Формальный |
| Res | Byte | Возвращаемый приоритет оператора | Формальный |

### Структура данных алгоритма NumbToMatr (Numb, Res)

Таблица – Структура данных алгоритма NumbToMatr (Numb, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Numb | Real | Число, которое необходимо перевести в матрицу | Формальный |

Продолжение таблицы 10

|  |  |  |  |
| --- | --- | --- | --- |
| Res | TMatrix | Возвращаемая матрица 1x1 | Формальный |

### Структура данных алгоритма SumMatrixes (FirstMatr, SecondMatr, Res)

Таблица - Структура данных алгоритма SumMatrixes (FirstMatr, SecondMatr, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i, j | Integer | Счетчики циклов | Локальный |
| FirstMatr | TMatrix | Первая передаваемая матрица | Формальный |
| SecondMatr | TMatrix | Вторая передаваемая матрица | Формальный |
| Res | TMatrix | Возвращаемая матрица – результат сложения двух матриц | Формальный |

### Структура данных алгоритма SubMatrixes (FirstMatr, SecondMatr, Res)

Таблица – Структура данных алгоритма SubMatrixes (FirstMatr, SecondMatr, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i, j | Integer | Счетчики циклов | Локальный |
| FirstMatr | TMatrix | Первая передаваемая матрица | Формальный |
| SecondMatr | TMatrix | Вторая передаваемая матрица | Формальный |
| Res | TMatrix | Возвращаемая матрица – результат разности двух матриц | Формальный |

### Структура данных алгоритма MultMatrixes (FirstMatr, SecondMatr, Res)

Таблица – Структура данных алгоритма MultMatrixes (FirstMatr, SecondMatr, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i, j | Integer | Счетчики циклов | Локальный |
| FirstMatr | TMatrix | Первая передаваемая матрица | Формальный |
| SecondMatr | TMatrix | Вторая передаваемая матрица | Формальный |
| Res | TMatrix | Возвращаемая матрица – результат произведения двух матриц | Формальный |

### Структура данных алгоритма MultConstMatrix (Numb, FirstMatr, Res)

Таблица – Структура данных алгоритма MultConstMatrix (Numb, FirstMatr, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i, j | Integer | Счетчики циклов | Локальный |
| Numb | Real | Передаваемое число | Формальный |
| FirstMatr | TMatrix | Передаваемая матрица | Формальный |
| Res | TMatrix | Возвращаемая матрица – результат произведения числа на матрицу | Формальный |