

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

КАЛЬКУЛЯТОР

БГУИР КП 1-40 02 01 323 ПЗ

Студент: гр. 250503 Патюпин М.С.

Руководитель: ассистент Богдан Е.В.

МИНСК 2023

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой

(подпись)

2023 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту Патюпину Михаилу Сергеевичу

1. Тема проекта Калькулятор
2. Срок сдачи студентом законченного проекта 15 декабря 2023 г.
3. Исходные данные к проекту: Программа написана в помощь студентам первого курса в проверке вычислений по математическому блоку учебной программы.
4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Лист задания.

Введение.

1. Обзор источников.

1.1. Обоснование разработки.

1.2. Обзор аналогов.

1.3. Обзор методов и алгоритмов решения поставленной задачи.

2. Функциональное проектирование.

2.1. Структура входных и выходных данных.

2.2. Разработка диаграммы классов.

2.3. Описание классов.

3. Разработка программных модулей.

3.1. Разработка схем алгоритмов.

3.2. Разработка алгоритмов.

4. Результаты работы.

5. Литература

Заключение

Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

3. Схема алгоритма `action scal()`.

Е.В. Богдан

_____ М.С. Патюпин
(дата и подпись студента)

СОДЕРЖАНИЕ

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ	5
ВВЕДЕНИЕ.....	6
1 ОБЗОР ИСТОЧНИКОВ	7
1.1 Обоснование разработки	7
1.2 Обзор аналогов	7
1.3 Обзор методов и алгоритмов решения поставленной задачи	8
2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ	9
2.1 Структура входных и выходных данных.....	9
2.2 Разработка диаграммы классов	10
2.3 Описание классов.....	10
3 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ	18
3.1 Разработка схем алгоритма работы приложения.....	18
3.2 Алгоритм вычисления детерминанта.....	18
3.3 Алгоритм нахождения ранга матрицы	19
4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	20
4.1 Описание файлов приложения	20
4.2 Руководство по использованию	22
ЗАКЛЮЧЕНИЕ	31
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	32
ПРИЛОЖЕНИЕ А.	33
ПРИЛОЖЕНИЕ Б.....	34
ПРИЛОЖЕНИЕ В	35
ПРИЛОЖЕНИЕ Г	36
ПРИЛОЖЕНИЕ Д	109

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Целью данного проекта является создание интуитивно понятного и удобного в использовании инструмента, который поможет студентам в их учебном процессе. В рамках этой работы предлагается разработать приложение “Калькулятор” с использованием фреймворка Qt.

В приложении должен быть реализован функционал для выполнения следующих задач в области векторной и линейной алгебры и сохранения ваших действий, и в предусмотренных случаях хода решения в файл:

- Определение ортогональности или коллинеарности двух и более векторов
- Вычисление скалярного произведения двух и более векторов
- Вычисление векторного произведения двух векторов
- Сложение и вычитание матриц
- Умножение матриц
- Вычисление детерминанта(определителя) матрицы
- Транспонирование матрицы
- Нахождение ранга матрицы
- Решение систем линейных уравнений

ВВЕДЕНИЕ

В современном мире, где технологии развиваются с беспрецедентной скоростью, образование становится все более цифровым. Студенты все чаще используют цифровые инструменты для улучшения своего обучения и повышения эффективности. Один из таких инструментов - это калькулятор.

Калькулятор - это универсальный инструмент, который может быть использован в различных областях, начиная от простых арифметических операций до сложных вычислений в области векторной, линейной алгебры, начала анализа. Однако, большинство существующих калькуляторов ограничены базовыми функциями и не предоставляют возможности для выполнения более сложных операций, которые часто требуются студентам.

В этой работе представлен проект калькулятора, который был разработан специально для студентов первого курса. Этот калькулятор предоставляет возможность работать в пределах линейной алгебры, что позволяет использовать его при подготовке к таким учебным дисциплинам как “Линейная алгебра и алгебраическая геометрия” и “Специальные математические методы и функции”. Кроме того, он обладает функцией сохранения хода решения в файл с объяснениями, что может быть очень полезно для студентов при изучении нового материала.

1. ОБЗОР ИСТОЧНИКОВ

1.1 Обоснование разработки

Для того, чтобы приступить к разработке, было необходимо ознакомиться с основными подходами к реализации математических вычислений на компьютере, изучить основные концепции уже существующих графических интерфейсов калькуляторов для создания собственного аналога, ничем не уступающего вышеупомянутому. Требовалось изучить основные принципы работы с числами: способы хранения значений, методы обработки ошибок и исключений. Необходимо иметь четкое представление о том, как хранятся и обрабатываются числовые значения, для того чтобы любая операция была корректной. Также было необходимо хорошо владеть языком программирования C++, для реализации корректной передачи данных между пользователем, графическим интерфейсом, который преимущественно написан с помощью декларативного языка программирования QML и самими алгоритмами, реализованными на C++, в то же время и для максимально эффективного использования ресурсов машины. Для написания грамотного кода требовалось ознакомиться с так называемыми “соглашениями” по написанию кода на C++ и QML, чтобы возможные расширения программы или ее рефакторинг не вызывали затруднений при реализации.

1.2 Обзор аналогов

В сфере пользовательских вычислений существует множество приложений, имеющих свои преимущества: HiPER Scientific Calculator, Wolfram Alpha/Mathematica, GeoGebra и многие другие. Все эти приложения – серьезные конкуренты, и поэтому, для того чтобы сделать данный проект конкурентно-способным, необходимо в первую очередь реализовать функционал, объединяющий уже существующие аналоги. Для этого выделим основные возможности конкурентов:

1. Направленный функционал.
2. Возможность экспортировать решение или входные/выходные данные.
3. Поддержка на компьютере с операционной системой Windows.

Крайне сложно перечислить все существующие на данный момент приложения: число их растет постоянно, а отличаются они тем, что стараются обогнать конкурентов, углубляясь в более узкие отрасли работы. Но все они, при этом, имеют и общий функционал, от которого и будет отталкиваться данный проект.

1.3 Обзор методов и алгоритмов решения поставленной задачи

Для написания курсового проекта был выбран язык программирования C++. Это компилируемый, мультипарадигмальный язык общего назначения с поддержкой процедурного программирования (возможность реализации на чистом C), обобщенного программирования (шаблоны) и объектно-ориентированного (классы и их объекты). C++ широко используется для разработки программного обеспечения, являясь при этом одним из самых популярных языков программирования в мире. На этом языке можно разработать как простейшие консольные приложения, так и целые операционные системы и драйверы.

Разработка курсового проекта строилась на основе парадигмы объектно-ориентированного программирования, которая подразумевает представление всей программы в виде совокупности взаимодействующих между собой объектов, которые являющихся экземплярами некоторых абстрактных сущностей – классов. Сами классы в свою очередь обладают собственными функциями, называемыми методами.

Широко использовались возможности языка C++, такие как работа с указателями и динамической памятью, хранение объектов в стандартных контейнерах библиотеки STL, использование абстрактных классов для доступа к классам потомкам с помощью полиморфизма.

Создание проекта выполнялось в кроссплатформенном фреймворке Qt версии 11.0.2. Данный фреймворк позволяет упростить работу с реализацией графического интерфейсом за счет наличия метаобъектного компилятора, обеспечивающего механизм сигналов/слотов, когда одни классы способны испускать сигналы при определенных условиях (например, при взаимодействии с пользователем), а другие перехватывать их и выполнять некоторые предопределенные операции (слоты). Для сборки проекта был использован компилятор MinGW версии 6.3.0

2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

2.1 Структура входных и выходных данных

Программа предназначена для выполнения различных математических операций. Ниже приведено описание структуры входных и выходных данных для каждой из этих операций:

- Определение ортогональности или коллинеарности векторов:
Входные данные – количество векторов, массив
Выходные данные – результат true/false
- Вычисление скалярного произведения двух и более векторов:
Входные данные – количество векторов, массив
Выходные данные – результат (число)
- Вычисление векторного произведения двух векторов:
Входные данные – массивы
Выходные данные – результат (массив)
- Сложение и вычитание матриц:
Входные данные – размерность матриц, массивы
Выходные данные – результат (размерность, массив)
- Умножение матриц:
Входные данные – размерность матриц, массивы
Выходные данные – результат (размерность, массив)
- Вычисление детерминанта(определителя) матрицы:
Входные данные – размерность матрицы, массив
Выходные данные – результат (число)
- Транспонирование матрицы:
Входные данные – размерность матрицы, массив
Выходные данные – результат (размерность, массив)
- Нахождение ранга матрицы:
Входные данные – размерность матрицы, массив
Выходные данные – результат (число)
- Решение систем линейных уравнений (далее СЛАУ) с помощью метода Гаусса:
Входные данные – размерность матриц, массивы
Выходные данные – результат (размерность, массив)
- Алгоритм рота:
Входные данные – набор L, N (массивы)
Выходные данные – результат (массив)

2.2 Разработка диаграммы классов

Диаграмма классов - это инструмент в языке моделирования UML, который используется для визуализации структуры классов в системе, их атрибутов, методов, интерфейсов и отношений между ними. Эта диаграмма обычно применяется при проектировании архитектуры, документировании системы, уточнении требований, а также для поддержки системы.

Диаграмма классов иллюстрирует модели данных даже для очень сложных информационных систем. На ней представлены в рамках, содержащих три компонента:

- В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом;
- В средней части перечислены атрибуты (поля) класса;
- В нижней части перечислены методы класса.

Диаграмма классов служит для визуализации статического представления системы, представляя различные аспекты приложения. Она представляет собой графическое представление статического представления системы и представляет различные аспекты приложения и также может включать в себя классы, их атрибуты, методы и отношения между классами, такие как наследование, агрегация, ассоциация, множественность ассоциации, обобщение, зависимость, использование, реализация и композиция.

Диаграмма классов данной работы показана в приложении А.

2.3 Описание классов

2.3.1 Класс `MainWindow`

`MainWindow` – представляет класс основного окна, который является скелетом программы и непосредственно отвечает за взаимодействие с пользователем.

Описание полей класса:

`Ui::MainWindow *ui` – основное окно, на котором располагается базовый интерфейс необходимый для взаимодействия пользователя с программой.

`UImatrix duo_matrix` – экземпляр класса, отвечающего за вычисления с матрицами, а также хранения данных о них.

`uivector duo_vec` – экземпляр класса, отвечающего за вычисления с матрицами, а также хранения данных о них.

`uialgrot rot_main` – экземпляр класса, отвечающего за работу алгоритма извлечения Рота.

Описание методов:

`MainWindow(QWidget *parent = nullptr)` - конструктор класса. Служит для создания окна программы и инициализации интерфейса.

`~MainWindow()` - деструктор класса. Служит для удаления окна программы. Это важно для корректной работы приложения и эффективной работы с памятью устройства.

`void on_menu_button_clicked()` – метод реагирующий на нажатие пользователем кнопки “menu” он отвечает за открытие и закрытие шторки с разделами калькулятора.

`void on_button_matrix_add_clicked()` - метод реагирующий на нажатие пользователем кнопки “matrix addition”. Он в `QQuicWidget` открывает окно `add_matrix.qml` в котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса отвечающего за работу с матрицами - `duo_matrix` в него.

`void on_button_matrix_sub_clicked()` - метод реагирующий на нажатие пользователем кнопки “matrix subtraction”. Он в `QQuicWidget` открывает окно `sub_matrix.qml` в котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса отвечающего за работу с матрицами - `duo_matrix` в него.

`void on_button_matrix_mul_clicked()` - метод реагирующий на нажатие пользователем кнопки “matrix multiplication”. Он в `QQuicWidget` открывает окно `mul_matrix.qml` в котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса отвечающего за работу с матрицами - `duo_matrix` в него.

`void on_button_matrix_transpore_clicked()` - метод реагирующий на нажатие пользователем кнопки “transpose matrix”. Он в `QQuicWidget` открывает окно `tra_matrix.qml` в котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса отвечающего за работу с матрицами - `duo_matrix` в него.

`void on_button_matrix_rnk_clicked()` - метод реагирующий на нажатие пользователем кнопки “rank of matrix”. Он в `QQuicWidget` открывает окно `rnk_matrix.qml` в котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса отвечающего за работу с матрицами - `duo_matrix` в него.

`void on_button_matrix_clay_clicked()` - метод реагирующий на нажатие пользователем кнопки “sustem of linear equations”. Он в `QQuicWidget` открывает окно `sly_matrix.qml` в котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса отвечающего за работу с матрицами - `duo_matrix` в него.

`void on_button_rot_clicked()` - метод реагирующий на нажатие пользователем кнопки “algorithm rota”. Он в `QQuicWidget` открывает окно `rot.qml` в котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса отвечающего за работу алгоритма - `rot_main` в него.

`void on_button_about_clicked()` - метод реагирующий на нажатие пользователем кнопки “about”. Он в QQuicWidget открывает окно `about.qml` в котором содержится информация о разработчике приложения.

`void on_new_window_clicked()` - метод реагирующий на нажатие пользователем кнопки “new window”. Который запускает новый процесс, выполняя исполняемый файл приложения в фоновом режиме.

`void on_open_log_clicked()` - метод реагирующий на нажатие пользователем кнопки “open log file”. С помощью системной команды открывает файл логов в `notepad`.

`void on_button_matrix_det_clicked()` - метод реагирующий на нажатие пользователем кнопки “determinant”. Он в QQuicWidget открывает окно `det_matrix.qml` котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса, отвечающего за работу с матрицами - `duo_matrix` в него.

`void on_button_dec_clicked()` - метод реагирующий на нажатие пользователем кнопки “orthogonal/collinear”. Он в QQuicWidget открывает окно `dec_vec.qml` котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса, отвечающего за работу с векторами - `duo_vec` в него.

`void on_button_scal_clicked()` - метод реагирующий на нажатие пользователем кнопки “scalar multiplication”. Он в QQuicWidget открывает окно `scal_vec.qml` котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса, отвечающего за работу с векторами - `duo_vec` в него.

`void on_button_vec_clicked()` - метод реагирующий на нажатие пользователем кнопки “vector multiplication”. Он в QQuicWidget открывает окно `dec_vec.qml` котором прописан интерфейс для взаимодействия с пользователем, и передает адрес экземпляра класса, отвечающего за работу с векторами - `duo_vec` в него.

2.3.2 Класс `log`

`log` – базовый класс, использующийся для записи в текстовый файл `log_file.txt` производимые вычисления в хронологическом порядке, или действия.

Описание полей класса:

`std::ofstream f_log` – объект потока вывода файла.

Описание методов класса:

`void f_del_data()` – метод для очистки содержимого файла.

`void f_log_vec(const std::string& str, const std::vector<std::string>& vec)` – метод принимающий текстовую строку, и вектор типа `string`, а после в соответствующем порядке записывает их в файл.

`void f_log_vec(const std::string& str, const std::vector<double>& vec)` – метод принимающий текстовую строку, и вектор типа `double`, а после в соответствующем порядке записывает их в файл.

`void f_log_text(const std::string& str)` – метод принимающий строку и записывающий ее в файл.

`void f_log_int(int i)` – метод принимающий переменную типа `int` и записывающий ее в файл.

2.3.3 Класс `matrix`

`matrix` – базовый класс матрицы, содержащий конструктор `matrix()` для инициализации полей класса.

Описание полей класса:

`std::vector<double> mat_data` – динамический массив элементов типа `double`, используется для хранения данных матрицы.

`int mat_cols` – поле для хранения информации о количестве столбцов в матрице.

`int mat_rows` – поле для хранения информации и количестве строк в матрице.

2.3.4 Класс `UImatrix`

`UImatrix` – производный класс от классов `matrix` и `log`, и базового класса `Qt Framework`. Что обеспечивает ему возможности для объектов в `Qt`, включая систему сигналов и слотов – которая и использовалась для обмена информацией с пользовательским интерфейсом.

Описание полей класса:

`matrix mat1` – объект типа `matrix` для хранения входных данных от пользователя.

`matrix mat2` – объект типа `matrix` для хранения входных данных от пользователя.

`matrix result` – объект типа `matrix` для хранения результата математической операции.

Описание методов класса:

`UImatrix(QObject *parent = nullptr)` - конструктор класса `UImatrix`. Он принимает указатель на родительский объект `QObject` и используется для инициализации объекта `UImatrix`.

`input_matrix1(QVariant data, int cols, int rows)` - слот для инициализации поля `matrix mat1`. Он принимает данные матрицы в виде обобщенного типа данных `Qt - QVariant` и размерность матрицы.

`input_matrix2(QVariant data, int cols, int rows)` - слот для инициализации поля `matrix mat2`. Он принимает данные матрицы в виде обобщенного типа данных `Qt - QVariant` и размерность матрицы.

`action_add()` - метод для выполнения операции сложения матриц. Он складывает матрицы `mat1` и `mat2` и сохраняет результат операции в `result`.

`action_sub()` - метод для выполнения операции вычитания матриц. Он вычитает матрицу `mat2` из матрицы `mat1` и сохраняет результат операции в `result`.

`action_mul()` - метод для выполнения операции умножения матриц. Он умножает матрицу `mat1` на матрицу `mat2` и сохраняет результат в `result`.

`action_tra()` - метод для выполнения операции транспонирования матрицы. Он транспонирует матрицу `mat1` и сохраняет результат в `result`.

`void UImatrix::action_rnk()` - метод находит ранг матрицы `mat1`. Алгоритм идентифицирует ненулевые строки и столбцы в матрице и преобразует матрицу в ступенчатый вид. Результат сохраняется в `result`.

`void UImatrix::action_sly()` - метод для решения системы линейных уравнений с помощью метода Гаусса.

`void UImatrix::action_det()` - метод для вычисления определителя матрицы `mat1`, с помощью разложения по строке.

`void UImatrix::log_start(const std::string& str)` - метод принимает название операции, используется для записи в файл начала вычислений, а также двух исходных матриц.

`void UImatrix::log_start_single(const std::string &str)` - метод принимает название операции, используется для записи в файл начала вычислений, а также единственной исходной матрицы.

`void UImatrix::log_end(const std::string &str)` - метод принимающий название операции, используется для записи результата операции, а также обозначения ее корректного завершения.

Описание сигналов класса:

`void sendMatrix(QVariant data, int cols, int rows)` - сигнал, для отправки матрицы в пользовательский интерфейс.

2.3.5 Класс `algrot`

`algrot` - производный класс от класса `log` - служащий для сохранения хода решения, и промежуточных результатов. Класс используется для минимизации булевой функции.

Описание полей класса:

`std::vector<std::string> Ai` - вектор для хранения множества новых кубов полученных на шаге умножения.

`std::vector<std::pair<std::string, int>> Ci` - вектор для хранения множества кубов, к которым применяется операция умножения для образования кубов большей размерности.

`std::vector<std::string> Z` - вектор для хранения множества простых импликант.

`std::vector<std::string> Bi` - вектор для хранения множества, получаемое из множества `Ci` удалением из него полученных на этом шаге простых импликант `Z`.

`std::vector<std::pair<std::string, std::string>> Lex` - вектор для хранения наборов которые могут быть L-экстремалами.

`std::vector<std::string> E` - вектор для хранения L-экстремалей.

`std::vector<std::string> NotCoverZ` - вектор для хранения наборов которые не покрыты множеством `Z`.

`log log` - объект класса `log`, предназначается для записи промежуточных результатов, а также решения в файл.

`std::vector<std::string> L` - вектор для хранения единичных кубов, введёнными пользователем.

`std::vector<std::string> N` - вектор для хранения безразличных кубов, введёнными пользователем.

`std::vector<std::vector<std::string>> Fmin` - вектор для хранения вектора элементарных конъюнций.

Описание методов класса:

`algrot(const std::vector<std::string> &L_, const std::vector<std::string> &N_)` - метод инициализирующий поля класса введёнными пользователем наборами `L` и `N`.

`void AiOst()` - операция получения множества `Ai`, то есть `*`.

`void inpCi()` - образование множества `Ci`.

`void correctAi()` - метод, удаляющие повторяющиеся элементы из `Ai`.

`void resBi()` - формирование множества `Bi`.

`void resZ()` - формирование множества `Z`.

`void correctZ()` - метод, удаляющий повторяющиеся наборы из `Z`.

`void resCi()` - формирование множества `Ci`.

`void canBeLext()` - метод для поиска возможных L-экстремалей из множества `Z`.

`void findLExtr()` - поиск L-экстремалей.

`void CorrectZ_forVetvl()` - алгоритм ветвления

`void NotcoverZ()` - поиск кубов не покрывающих другие кубы.

`void resFmin(bool alg)` - метод для нахождения минимального покрытия множества `L` множеством `Z`.

`void getE()` - метод для записи L-экстремалей в файл.

`void getNoCoverZ()` - метод для записи в файл множества непокрытых импликант.

`void getCi()` - метод для записи в файл множеств `Ci`.

`void getZ()` - метод для записи в файл множества `Z`.

`void getAi()` - метод для записи в файл множеств `Ai`.

`void getBi()` - метод для записи в файл множества `Bi`.

`void getN()` - метод для записи в файл не обязательного множества безразличных наборов.

void getL() -- метод для записи в файл множества единичных наборов.
 void getSetsMbLex() – метод для записи в файл простых импликант которые могут быть L-экстремалиями.
 void getFmins() – метод для записи в файл минимальной функции.
 void AlgRot() – алгоритм Рота, поэтапно.
 void clear() – очистка полей класса.

2.3.6 Класс `uialgrot`

`Uialgrot` - производный класс от классов `rot` и базового класса `Qt Framework`. Что обеспечивает ему возможности для объектов в `Qt`, включая систему сигналов и слотов – которая и использовалась для обмена информацией с пользовательским интерфейсом.

Описание полей класса:

`std::vector<std::string> L_my` – массив в который будут передаваться данные из пользовательского интерфейса.

`std::vector<std::string> N_my` – массив в который будут передаваться данные из пользовательского интерфейса.

Описание методов класса:

void input_L_N(QVariantList L, QVariantList N) – слот, для принятия классом данных от интерфейса.

void action_rot() – метод, который может быть вызван в файле `rot.qml` для старта алгоритма Рота.

void prepare_f() – метод, для подготовки данных, и отправки сигнала `send_f`.

Описание сигналов класса:

void send_f(QVariantList data) – сигнал, позволяющий подключиться другому объекту и получать данные через него.

2.3.6 Класс `vector`

`vector` – базовый класс для `uivector`. В классе отсутствует поле предназначенного для содержания информации о количестве векторов в каждом массиве, так как предполагается, что при необходимости сохранить несколько векторов в одном массиве можно записав их последовательно, и так как у каждого из них три координаты.

Описание полей класса:

`std::vector<double> vec1` – первый массив для хранения вектора.

`std::vector<double> vec2` – второй массив для хранения вектора.

`std::vector<double> res` – массив для хранения результата операции.

`struct point {double x, y, z;}` – структура для хранения координат точки.

Описание методов класса:

`vector()` – метод для инициализации полей класса.

2.3.7 Класс `uivector`

`uivector` - производный класс от классов `vector`, `log`, и базового класса `Qt Framework`. Что обеспечивает ему возможности для объектов в `Qt`, включая систему сигналов и слотов – которая и использовалась для обмена информацией с пользовательским интерфейсом.

Описание полей класса:

`std::vector<double> vec_max` – дополнительный массив для хранения векторов.

Описание методов класса:

`explicit uivector(QObject *parent = nullptr)` -явный конструктор класса.

`void input_vec1(QVariant data)` – метод для заполнения массива `vec1` пользовательскими данными.

`void input_vec2(QVariant data)` – метод для заполнения массива `vec2` пользовательскими данными.

`void input_vec_max(QVariant data)` – метод для заполнения массива `vec_max` пользовательскими данными.

`void action_kol()` – метод для определения коллинеарности и ортогональности векторов.

`void action_scal()` – метод для выполнения скалярного умножения векторов.

`void action_vec()` - метод для выполнения векторного умножения векторов.

`void prepareVec()` – метод, для передачи вектора `res` в виде параметра типа `QVariant` через сигнал `sendVec()`.

Описание сигналов класса:

`void sendVec(QVariant data)` - сигнал, позволяющий подключиться другому объекту и получать данные через него.

3 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

3.1 Разработка схем алгоритма работы приложения

При выборе пользователем пункта из меню вызывается индивидуальный слот-обработчик – процедура, которая открывает в виджете стороннюю с интерфейсом, и подключает экземпляр необходимого в работе класса к ней.

Далее будут описаны основные методы, которые используются в данном приложении.

Метод `calculateDeterminant(int rows, int cols, const std::vector<double>& matrix)` предназначен для вычисления определителя матрицы. Этот метод способен детерминант не зависимо от размера матрицы, но при создании графического интерфейса умышлено было установлено ограничение в 5 столбцов, строк.

Метод `action_vec()` предназначен для вычисления векторного произведения двух векторов.

Схема метода `action_mul()` для нахождения произведения матриц показана в приложении Б.

Схема метода `action_scal()` для скалярного произведения векторов показана в приложении В.

3.2 Алгоритм вычисления детерминанта

Шаг 1. Проверка размера матрицы. Если матрица содержит только один элемент, то возвращаем его. Алгоритм завершен.

Шаг 2. Инициализация `determinant` нулем, и `sign` единицей.

Шаг 3. Начало внешнего цикла по столбцам матрицы, перебираем столбцы поочередно.

Шаг 4. Создание массива `submatrix` для хранения подматрицы, размерность подматрицы на единицу меньше размерности исходной.

Шаг 5. Начало внутренних циклов по строкам, и столбцам исходной матрицы, в переборе по каждой строке перебираем все столбцы.

Шаг 6. Добавление в `submatrix` каждого элемента основной матрицы, за исключением той строки и того столбца где мы находимся.

Шаг 7. Если не во внутреннем цикле не пройдены все строки, переходим к шагу 6.

Шаг 8. Обновление определителя путем добавления произведения текущего знака, текущего элемента матрицы и определителя подматрицы (рекурсивный вызов).

Шаг 9. Изменение знака на противоположный.

Шаг 10. Если во внешнем цикле не пройдены все столбцы, переход к шагу 4.

Шаг 11. Возвращение `determinant`.

3.3 Алгоритм нахождения ранга матрицы

Шаг 1. Вызов методов совершающих логирование. В файл записываем название операции, исходные вектора `vec1`, `vec2`.

Шаг 2. Очищаем вектор результата, с помощью функции `clear()`.

Шаг 3. Добавление в конец вектора нового элемента, инициализация элемента разностью произведений второго элемента `vec1`, третьего элемента `vec2` и третьего элемента `vec1`, второго элемента `vec2`.

Шаг 4. Добавление в конец вектора нового элемента, инициализация элемента разностью произведений третьего элемента `vec1`, первого элемента `vec2` и первого элемента `vec1`, третьего элемента `vec2`.

Шаг 5. Добавление в конец вектора нового элемента, инициализация элемента разностью произведений первого элемента `vec1`, второго элемента `vec2` и второго элемента `vec1`, первого элемента `vec2`.

Шаг 6. Вызов метода `prepareVec()` совершающий отправку вектора `res` в `qml` интерфейс для отображения результата пользователю.

Шаг 7. Вызов методов совершающих логирование. В файл записываем результат, и сообщение об успешном завершении алгоритма.

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Результатом выполнения курсового проекта стало консольное приложение для платформы Windows.

4.1 Описание файлов приложения

В ходе разработки программного приложения, были созданы файлы нескольких разрешений:

- .cpp – для реализаций функций и классов, содержат логику и алгоритмы, которые выполняются при работе программы.
- .h – заголовочные файлы, содержат объявления функций, методов, классов, структур описанных в одноименных файлах.cpp.
- .qml – файлы на декларативном языке Qt Meta-Object Language, в основании которого лежит среда JavaScript. В этих файлах описан графический интерфейс отображающийся в поле QQuickWidget.
- .ui – файл интерфейса пользователя в формате eXtensible Markup Language, созданный с использованием инструмента Qt Designer

Ниже приведен список файлов и их описание:

- about.qml – файл, с графическим интерфейсом страницы содержащую данных о разработчике.
- add_matrix.qml – файл, содержащий интерфейс для ввода пользователем двух матриц размерности от 1 до 5. Также и вывода результата.
- algrot.cpp – файл, содержащий все методы и функции необходимые для выполнения алгоритма Рота.
- algrot.h – файл, содержащий объявление класса rot.
- dec_vec.qml – файл, содержащий интерфейс для ввода пользователем нескольких векторов и последующего вывода результата скалярного умножения векторов.
- det_matrix.qml – файл, содержащий интерфейс для ввода пользователем матрицы размерности от 1 до 5 и вывода определителя.
- general.qml - файл с графическим интерфейсом страницы приветствия.
- log.h - заголовочный файл, содержащий объявление класса log.
- main.cpp - файл, содержащий основную функцию программы – main, инициализация компонентов Qt.
- mainwindow.cpp - файл, содержащий функции, реагирующие на воздействие пользователем на кнопки главного интерфейса.
- mainwindow.h - файл, содержащий объявление основного класса программы MainWindow.
- mainwindow.ui - файл, главного окна приложения.
- matrix.cpp - файл, содержащий реализацию методов класса matrix.

- `matrix.h` - файл, содержащий объявление класса `matrix`.
- `mul_matrix.qml` - файл, содержащий интерфейс для умножения матриц: ввода размерности, самих данных, просмотра результата.
- `rnk_matrix.qml` - файл, содержащий интерфейс для ввода пользователем матрицы и вывода ее ранга.
- `rot.qml` - файл, содержащий интерфейс для ввода пользователем множества единичных наборов, а также множества безразличных.
- `scal_vec.qml` - файл, содержащий интерфейс для ввода пользователем нескольких векторов и вывода результата их скалярного умножения.
- `sly_matrix.qml` - файл, содержащий интерфейс для ввода системы линейных уравнений.
- `sub_matrix.qml` - файл, содержащий интерфейс для вычитания одной матрицы из другой.
- `tra_matrix.qml` - файл, содержащий интерфейс для ввода пользователем матрицы и вывода ее транспонированной матрицы.
- `uialgrot.cpp` - файл, содержащий реализацию класса `uialgrot`, отвечающего за взаимодействие с пользовательским интерфейсом для алгоритма Рота.
- `uialgrot.h` - файл, содержащий объявление класса `uialgrot`, отвечающего за взаимодействие с пользовательским интерфейсом для алгоритма Рота.
- `uimatrix.cpp` - файл, содержащий реализацию класса `uimatrix`, отвечающего за взаимодействие пользовательского интерфейса и класса `matrix`.
- `uimatrix.h` - файл, содержащий объявление класса `uimatrix`, отвечающего за взаимодействие пользовательского интерфейса и класса `matrix`.
- `uivector.cpp` - файл, содержащий реализацию класса `uivector`, отвечающего за взаимодействие пользовательского интерфейса и класса `vector`.
- `uivector.h` - файл, содержащий объявление класса `uivector`, отвечающего за взаимодействие пользовательского интерфейса и класса `vector`.
- `vec_vec.qml` - файл, содержащий интерфейс для ввода пользователем двух векторов и вывода результата их векторного умножения.
- `vector.cpp` - файл с реализацией методов класса `vector` предназначенных для выполнения функций с векторами.
- `vector.h` - файл с объявлением класса `vector` предназначенного для выполнения функций с векторами.

4.2 Руководство по использованию

При запуске программы пользователя встречает главное окно (рис. 4.1). В главном окне располагается 3 основные кнопки:

- menu – открытие меню-шторки для выбора категории вычислений.
- new window – открытие нового рабочего окна.
- open log file – открытие файла содержащего информацию действиях пользователя, производимых вычислений (их исходных данных и результата).

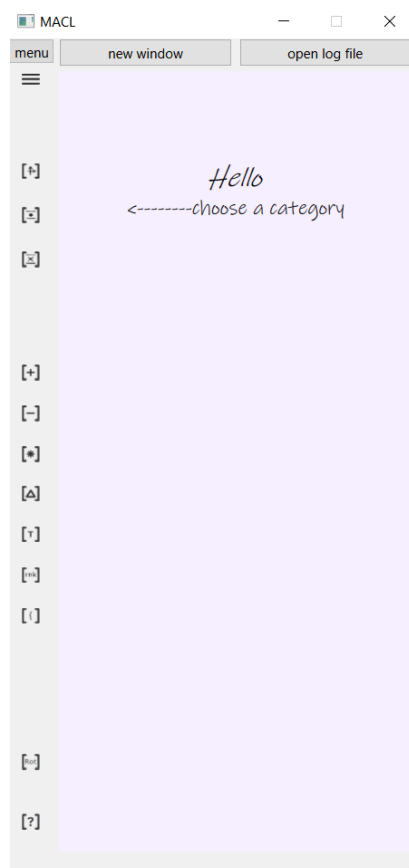


Рисунок 4.1 – Главное окно

Выбрав паркую кнопку – “menu” пользователю предлагается выбрать один из 11 вариантов вычислений в трёх категориях (рис. 4.2), а также просмотреть информацию о разработчике.

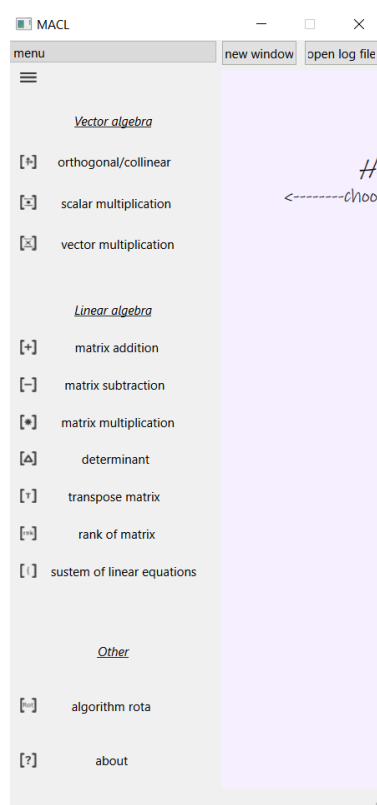


Рисунок 4.2

Выбрав первый пункт (“ortoganal/collinear”) из категории “Vector algebra”, пользователю предлагается ввести количество векторов, а также их значения для определения их относительного положения (рис. 4.3).

Рисунок 4.3

Выбрав второй пункт (“scalar multiplication”) из категории “Vector algebra”, пользователю предлагается ввести количество векторов, а также их значения (рис. 4.4) для нахождения их скалярного произведения.

Введите вектора:

1	2	3
4	5	6
7	8	9

Вычислить

Скалярное произведение = 270

Рисунок 4.4 – Нахождение скалярного произведения векторов

Выбрав последний пункт (“vector multiplication”) из категории “Vector algebra”, пользователю предлагается ввести два вектора для нахождения их векторного произведения (рис. 4.5).

Введите вектора:

2	-3	1
1	4	0

Вычислить

Векторное произведение = {-4; 1; 11}

Рисунок 4.5 – Нахождение векторного произведения векторов

Выбрав первый пункт (“matrix addition”) из категории “Linear algebra”, пользователю предлагается ввести размерность матриц, а также их значения для определения их суммы (рис. 4.6).

Размерность матриц:

2

2

Первая матрица:

11	12
21	22

Вторая матрица:

11	12
21	22

Вычислить

Результат сложения матриц:

22	24
42	44

Рисунок 4.6 - Сложение матриц

Выбрав второй пункт (“matrix subtraction”) из категории “Linear algebra”, пользователю предлагается ввести размерность матриц, а также их значения для определения их разности (рис. 4.7).

Размерность матриц:

1

3

Вторая матрица:

111	112	113
-----	-----	-----

Вторая матрица:

11	12	13
----	----	----

Вычислить

Результат вычитания матриц:

100	100	100
-----	-----	-----

Рисунок 4.7 – Разность матриц

Выбрав третий пункт (“matrix multiplication”) из категории “Linear algebra”, пользователю предлагается ввести размерность матриц, а также их значения для определения их произведения (рис. 4.8).

Размерность матриц:

2

3

1

Первая матрица:

1	2	3
4	5	6

Вторая матрица:

1
2
3

Вычислить

Результат сложения матриц:

14
32

Рисунок 5.8 – Умножение матриц

Выбрав четвертый пункт (“determinant”) из категории “Linear algebra”, пользователю предлагается ввести размерность матрицы, а также ее значения для определения детерминанта (рис. 4.9).

Размерность матрицы:

3

-1	3	2
2	8	1
2	2	4

Найти детерминант

Детерминант = -72

Рисунок 4.9 – Нахождение определителя матрицы

Выбрав пятый пункт (“transpose matrix”) из категории “Linear algebra”, пользователю предлагается ввести размерность матрицы, а также ее значения для нахождения транспонированной матрицы(рис. 4.10).

Размерность матрицы:

3

2

11	12
21	22
31	32

Транспонировать

11	21	31
12	22	32

Рисунок 4.10 – Транспонирование матрицы

Выбрав шестой пункт (“rank of matrix”) из категории “Linear algebra”, пользователю предлагается ввести размерность матрицы, а также ее значения для определения её ранга (рис. 4.11).

Размерность матрицы:

4

5

1	1	2	3	-1
2	-1	0	-4	-5
-1	-1	0	-3	-2
6	3	4	8	-3

Найти ранг

Ранг матрицы = 3

Рисунок 4.11 – Нахождение ранка матрицы

Выбрав седьмой пункт (“system of linear equations”) из категории “Linear algebra”, пользователю предлагается ввести количество уравнений и неизвестных, причем система должна являться квадратной, совместной и определенной, а также ее ввести значения при неизвестных для вычисления её решения (рис. 4.12).

Решение системы уравнений

3

2 1 -1 = 0

a1 3 4 = -6

1 a2 1 = 1

Вычислить

Решением являются значения a: 1; -2; 0

Рисунок 4.12 – Решение системы линейных уравнений

Выбрав первый пункт (“algorithm rota”) из категории “Other”, пользователю предлагается ввести количество единичных наборов, и также количество неопределенных. После – сами наборы (рис. 4.13).

Введите исходные данные

6

0001

0100

1100

1101

1010

0010

3

0101

0111

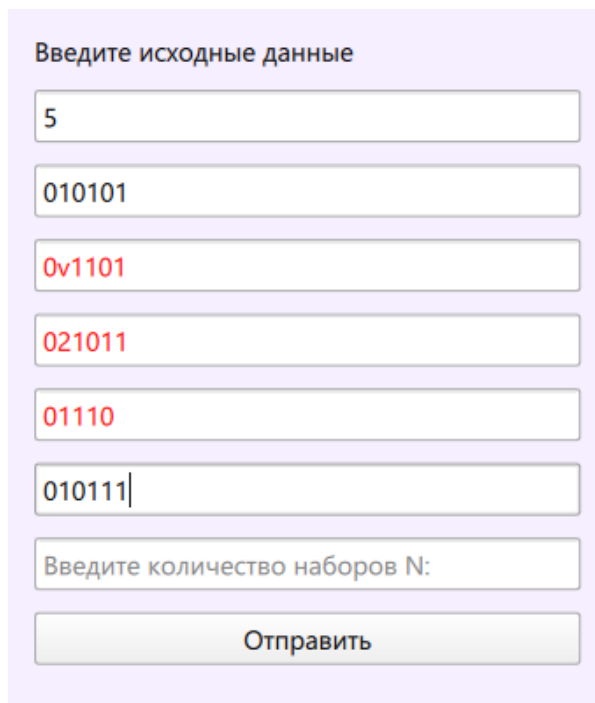
0110

Отправить

Фмднф = {0x01, x010, x10x}

Рисунок 4.13 – Алгоритм Рота

Подсвечивание наборов красным цветом (рис. 4.14) свидетельствует о неверно введенных данных, ввод анализируется на количество цифр в каждом наборе, на присутствие иных символов за исключением: “0”, “1”, “x”. Алгоритм может использоваться только один раз при каждом запуске программы.



Введите исходные данные

5

010101

0v1101

021011

01110

010111|

Введите количество наборов N:

Отправить

Рисунок 4.14 – Возможные ошибки при вводе

Выбрав второй пункт (“about”) из категории “Other”, пользователю представляется информация о разработчике, научном руководителе, и другая информация (рис. 4.15).

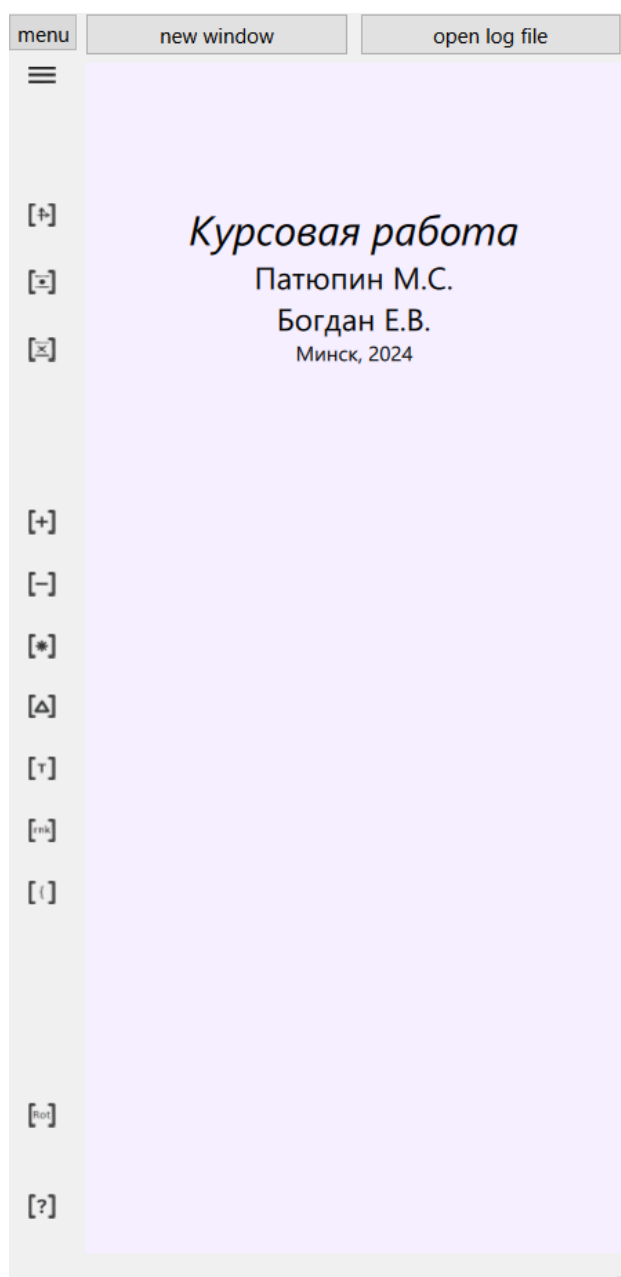


Рисунок 4.15 – Вкладка «about»

Выход из программы осуществляется общепринятыми в операционной системе windows - кнопкой “крестик” (рис. 4.16). Предоставляется возможность свернуть приложение, но не изменить его размер.



Рисунок 4.16 – Верхняя панель приложения

ЗАКЛЮЧЕНИЕ

Программа была разработана в соответствии с нуждами студентов обучающихся на первом курсе по специальности “Компьютерная инженерия”. Были изучены и отобраны наиболее часто встречающиеся задания в индивидуальных домашних заданиях[1] по предмету “Линейная алгебра и аналитическая геометрия”, алгоритмы были разработаны в соответствии учебником рекомендованным кафедрой высшей математики[2]. Дополнительно к заданию, в процессе проектирования программы был добавлен алгоритм извлечения Рота[3], рассматриваемый студентами во втором семестре по учебной дисциплине “Арифметические и логические основы вычислительной техники”.

В ходе выполнении работы, были получены знания и опыт использования фреймворка Qt для конструирования программы под операционную систему Windows, приложения Qt Designer для графического создания интерфейса пользователя. Был изучен язык разметки QML для реализации быстрой смены рабочего пространства, повышения читаемости кода, и точечного изменения. Научился проводить исследование целевой аудитории, для выявления задач которые необходимо реализовать в продукте.

При дальнейшей разработке проекта “Калькулятор” могут быть добавлены следующие функции:

- Расширение математических функций. Добавления таких глав высшей математики как “Дифференциальное исчисление функций одной переменной”, “Введение в анализ”, “Числовые ряды”, “Элементы теории функции комплексного переменного”.
- Расширение функций для самопроверки и подготовки студентов по дисциплине “Арифметические и логические основы вычислительной техники” , добавление алгоритмов умножения, деления, сложение точек с плавающей запятой с подробным объяснением хода решения.
- Добавление нового способа ввода. Используя распознавание текста по фотографии в автоматическом режиме определять к какому роду задач он может относиться, и импортировать данные. Добавление распознавания рукописного ввода[4].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Рябушко, А. П. Высшая математика : теория и задачи : учеб. Пособие. В 5ч. Ч. 1. Линейная и векторная алгебра. Аналитическая геометрия. Дифференциальное исчисление функций одной переменной / А. П. Рябушко, Т. А. Жур, 2-е изд. – Минск : Вышейшая школа, 2017. – 303 с. : ил.
- 2 Письменный, Д. Т. Конспект лекций по высшей математике: полный курс \ Д. Т. Письменный. – 17-е изд. – М.:АЙРИС-пресс, 2020. – 608 с.: ил.
- 3 Луцик Ю. А. Минимизация булевых функций [Электронный ресурс]. – Электронные данные. – Режим доступа: https://drive.google.com/drive/folders/1JjMgUx0fdXMi99A6wjjKv9bCJ_wW2KUf. – Дата доступа: 23.11.2022.
- 4 Можейко Д. О. Перцептрон. Распознавание слов. Курсовая работа : общие требования / Д. О. Можейко. – Минск : БГУИР, 2023.
- 5 Рожнова, Н. Г. Вычислительные машины, системы и сети. Дипломное проектирование : учебно-метод.пособие / Н. Г. Рожнова, Н. А. Искра, И. И. Глецевич. – Минск : БГУИР, 2014. – 96 с. : ил.
- 6 Бушкевич А. В. Конструирование программ и языки программирования: метод. К65 указания по курсовому проектированию для студ. Спец. 1-40 02 01 “Вычислительные машины, системы и сети” для всех форм обуч. / сост. А. В. Бушкевич, А. М. Ковальчук, И. В. Лукьянова. – Минск : БГУИР, 2010 – 30 с.: ил.
- 7 Шлее М. - Qt4. Профессиональное программирование на C+/ Шлее М. - Л.:Наука, 2013. - 770 с.
- 8 Дейтел, Х. Как программировать на C++ / Х. Дейтел, П. Дейтел. - М. : БИНОМ, 2001. - 1152 с.

ПРИЛОЖЕНИЕ А
(обязательное)
Диаграмма классов

ПРИЛОЖЕНИЕ Б
(обязательное)
Схема метода *action_mul()*

ПРИЛОЖЕНИЕ В
(обязательное)
Схема метода `action_scal()`

ПРИЛОЖЕНИЕ Г
(обязательное)
Код программы

ПРИЛОЖЕНИЕ Д
(обязательное)
Ведомость документов