**МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ**

**ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)**

Кафедра

«Автоматизированные системы управления»

Курсовая работа по дисциплине

«Программирование»

Вариант № 13

Тема:

**«Разработка программного приложения для прогнозирования изменений интенсивности дорожного движения»**

Выполнил: студент группы 2бАСУ1

Ручкин М. А.

Проверил: доцент каф. АСУ

Баринов К. А.

Москва 2024

Оглавление

[**1. Задание** 3](#_Toc187417622)

[**2. Назначение программы** 4](#_Toc187417623)

[**3. Руководство пользователя** 5](#_Toc187417624)

[**4. Выбор среды разработки** 14](#_Toc187417625)

[**5. Форматы данных** 15](#_Toc187417626)

[**5.1 Входные данные** 15](#_Toc187417627)

[**5.2 Выходные данные** 17](#_Toc187417628)

[**6. Структура программы** 20](#_Toc187417629)

[**6.1 Модуль <MyUnit>** 20](#_Toc187417630)

[**6.1.3.1 Класс оконной формы <TForm1>** 20](#_Toc187417631)

[**6.1.3.2 Методы класса формы** 22](#_Toc187417632)

[**6.1.4. Прочие методы** 24](#_Toc187417633)

[**7. Описание алгоритмов** 26](#_Toc187417634)

[**7.1 Блок-схема укрупненного алгоритма работы пользовательского интерфейса приложения** 26](#_Toc187417635)

[**7.2 void \_\_fastcall TForm1::OpenClick(TObject \*Sender)** 29](#_Toc187417636)

[**7.3 void \_\_fastcall TForm1::SaveClick(TObject \*Sender)** 29](#_Toc187417637)

[**7.4 void \_\_fastcall FormCloseQuery(TObject \*Sender, bool &CanClose)** 30](#_Toc187417638)

[**8. Текст программы с комментариями** 31](#_Toc187417639)

[**8.1 Заголовочный файл MyUnit.h** 31](#_Toc187417640)

[**8.2 Файл реализации MyUnit.cpp** 33](#_Toc187417641)

# **1. Задание**

**Вариант №13**

Для прогнозирования изменений интенсивности движения автотранспорта по дороге используется формула:

****, где

 - интенсивность движения в год *Т* (авт./сут);

 - интенсивность в первый год эксплуатации;

 - темп относительного прироста интенсивности в *t*-ый год эксплуатации дороги (%), , где

 - эмпирические коэффициенты, соответствующие некоторому первоначальному темпу относительного прироста интенсивности  (%) (например, ,  для  = 8%).

**Задания:**

1. Рассчитать изменение интенсивности движения по годам в интервале от  до с шагом . Результат напечатать в виде таблицы. Принять: , , , , авт./сут.
2. Рассчитать изменение интенсивности движения по годам в интервале от  до  для следующих наборов параметров:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | *a* | *b* |
| 1800 | 8% | 4.3 | 3.7 |
| 2100 | 8.7% | 4.6 | 3.91 |
| 2500 | 9.1% | 4.6 | 3.89 |
| 3000 | 9.41% | 4.73 | 4.01 |

Форму выходной таблицы с указанием всех данных разработать самостоятельно.

3. Таблицу задания 1 вывести в виде графика, реализовав этот алгоритм в виде процедурного блока.

# **2. Назначение программы**

Программа предназначена для автоматизации процесса прогнозирования изменений интенсивности дорожного движения по годам. Она позволяет рассчитывать ежегодные значения интенсивности на основе заданных параметров и отображать результаты в виде таблиц и графиков, что упрощает анализ и планирование транспортных потоков.

# **3. Руководство пользователя**

Начальный вид приложения:

(Рисунок 1)

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок

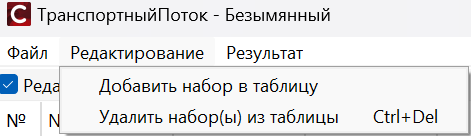
Подменю “Файл” (Рисунок 2) позволят создать новую таблицу, нажатием на меню “Новый”, открыть файл с заранее созданной таблицу “Открыть…”, сохранить текущую таблицу в тот же файл или создать новый “Сохранить”, и сохранить таблицу в любой удобный файл “Сохранить как…” в расширение “.tfi” (Trafic Flow Intensivity) и выйти из приложения “Выход”.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

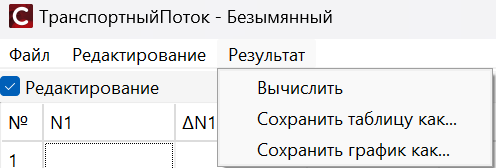
Рисунок

Подменю “Редактирование”(Рисунок 3) позволяет добавить в конец таблицы с исходными данными пустую строку “Добавить набор в таблицу”, удалить выделенные строки из таблицы с исходными данными “Удалить набор(ы) из таблицы”

****

Рисунок

Подменю “Результат”(Рисунок 4) позволяет вычилить данные и заполнить ими результирующую таюлицу и вывести данные на график “Вычислить”, а так же сохранить таблицу с даннми с расширением “.csv” “Сохранить таблицу как…” и график с расширением “.png” или “.emf” “Сохранить график как…”

****

Рисунок

Чтобы загрузить данные, необходимо выбрать в главном меню «Файл» -> «Открыть…». В диалогом окне, нужно выбрать файл с расширением “.tfi”(Рисунок 5).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок

Если файла не существует, то вам выведется сообщение об этом (Рисунок 6).

Изображение выглядит как текст, снимок экрана, Операционная система, дисплей

Автоматически созданное описание

Рисунок

После успешного открытия, в статус-баре появится сообщение об этом, а в заголовок приложения поместится имя вашего файла (Рисунок 7).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок

Чтобы сохранить набранные в таблице данные, необходим нажать на «Файл» -> «Сохранить» или на соответствующую кнопку на панели управления или на «Файл» -> «Сохранить как…» и в появившемся диалоговом окне (Рисунок 8) перезаписать существующий файл или создать новый.

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание**

Рисунок

После успешного сохранения, в статус-баре появится сообщение об этом (Рисунок 9).



Рисунок

Чтобы создать новый набор данных, необходимо нажать на «Файл» -> «Создать».

Кнопки на форме копируют часть функционала главного меню (Рисунок 10):  
“Вычислить” = Пункт главного меню “Результат”-“Вычислить”

“Добавить набор” = Пункт главного меню “Редактирование”-“Добавить набор в таблицу”

“Удалить набор(ы)” = Пункт главного меню “Редактирование”-“Удалить набор(ы) из таблицы ”

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок

При нажатии кнопки “Добавить набор” в конце таблицы добавляется пустая строка

Для удаления строки нужно выбрать любую ячейку строки и нажать на кнопку “Удалить набор(ы)”

Для удаления нескольких строк нужно включить галочку “Редактирование” и выделить наборы, которые вы желаете удалить (Рисунок 11).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок

После нажатия на кнопку удаления у вас появится окно для подтверждения вашего действия (Рисунок 12).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок

Для того, чтобы вычислить данные нужно нажать кнопку “Вычислить”. После успешного вычисления появятся все данные и сообщение в статус-баре (Рисунок 13)

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Рисунок

Для вычисления должны быть заполнены все поля ввода (Рисунок 14), конечный год должен быть больше начального (Рисунок 15), интервал между годами не должен равняться 0 (Рисунок 16), а также все данные должны быть числами (Рисунок 17, 18, 19, 20). Если что-то из этого не соблюдено, то в статус-баре будет сообщение об этом



Рисунок



Рисунок



Рисунок



Рисунок



Рисунок



Рисунок



Рисунок

Так же есть возможность установки опитимально ширины столбца таблицы при двойном щелчке по правому краю столбца (На примере столбец “a”) (Рисунок 21).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок

Чтобы сохранить вычисленные данные из таблицы нужно выбрать Пункт главного меню “Результат”-“Сохранить таблицу как… ” . В появившемся диалоговом окне (Рисунок 22) перезаписать существующий файл или создать новый

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок

Чтобы сохранить отрисованные данные с графика нужно выбрать Пункт главного меню “Результат”-“Сохранить график как… ” . В появившемся диалоговом окне (Рисунок 23) выбрать расширение “.png” или “.emf”, перезаписать существующий файл или создать новый.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок

# **4. Выбор среды разработки**

Для разработки приложения выбрана среда C++ Builder от Embarcadero RAD Studio благодаря её стабильности и широким функциональным возможностям. Среди ключевых преимуществ: обширная библиотека компонентов VCL для быстрой разработки интерфейсов, удобные инструменты для создания графиков и таблиц и поддержка пользовательских функций, что делает её оптимальной для выполнения данной курсовой работы.

# **5. Форматы данных**

## **5.1 Входные данные**

**Обоснование выбора двоичного формата файла:**

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Назначение** | **Диапазон значений** |
| N1 | Интенсивность в первый год эксплуатации | от 0 до 1000000 (авт./сут) |
| deltaN1 | Темп относительного прироста интенсивности в 1-ый год | от 0 до 100 (%) |
| a, b | эмпирические коэффициенты | от 0 до 100 |
| T0 | Начальный год | от 0 до 2025 |
| Tmax | Конечный год | от 1 до 3000 |
| deltaT | Интервал между годами | от 1 до 200 |

* **Эффективное хранение числовых данных:**

В двоичном файле каждая запись занимает строго определенное количество байт, что позволяет экономить место.

* + Объём занимаемый в двоичном формате(n – количество наборов):  
    N1 — 4 байта\*n (целое число);

deltaN1, a, b — по 4 байта\*n каждое (числа с плавающей точкой);

T0, Tmax — по 2 байта (целые числа);

deltaT — 1 байт (целое число).

**Итого для одной записи:** 16\*n+5 байт

* + В текстовом формате 1 символ кодируется 1 байтом + потребуется хотя бы один символ-разделитель между записями.   
    Средние значения занимаемые каждым полем:  
    N1 – 1-7 символов\*n  
    deltaN1 – 1-5 символов\*n  
    a, b – 3 символа\*n  
    T0, Tmax – 1-4 символов  
    deltaT – 1-3 символа  
    **Итого для одной записи:** 12\*n+5 – 22\*n+13 байт
* **Высокая скорость операций:**Отсутствие преобразований чисел в текст и обратно ускоряет чтение и запись. Фиксированный размер записей позволяет быстро переходить к нужным данным через позиционирование указателя.
* **Точность числовых расчетов:**В вашей задаче значения параметров a, b, ΔN1​ ​ являются числами с плавающей точкой, которые требуют точного представления. Текстовый формат может привести к потере точности из-за округления или ограничений при конверсии в строку. Двоичный формат полностью исключает эти проблемы, так как сохраняет данные в исходном виде.
* **Защита от непреднамеренных изменений**:
  + Текстовые файлы легко редактируются вручную. Пользователь может случайно удалить пробелы, изменить порядок полей или ввести недопустимые значения.
  + Двоичные файлы не предназначены для ручного редактирования, что минимизирует риск ошибок. Например, если кто-то попытается открыть двоичный файл и изменить содержимое, структура файла нарушится, что сразу будет обнаружено программой.

Таким образом, двоичный формат хранения данных обеспечивает экономичное использование памяти, высокую производительность и защиту от случайных изменений, что делает его подходящим решением для этой задачи.

**Структура двоичного файла:**

n шт.

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных, | Назначение |
| N1 | 4 байта, целое беззнак. | Начальная интенсивность движения N1​ (авт./сут.) |
| deltaN1 | 4 байта, с плав. точкой | Темп относительного прироста интенсивности ΔN1​ (%) |
| a | 4 байта, с плав. точкой | Коэффициент a, эмпирическое значение для расчета |
| b | 4 байта, с плав. точкой | |  | | --- | |  |  |  | | --- | | Коэффициент b, эмпирическое значение для расчета | |
| T0 | 2 байта, целое беззнак. | Начальный год T0 |
| Tmax | 2 байта, целое беззнак. | |  | | --- | |  |   Максимальный год Tmax |
| deltaT | 1 байт, целое беззнак. | Шаг ΔT (размер интервала) |

## **5.2 Выходные данные**

Для выходных данных в данной задаче предлагаются два формата файлов:

1. **CSV-файл** для сохранения результатов расчетов.
2. **PNG-файл** для сохранения графического представления данных в растровом формате.
3. **EMF-файл** для сохранения графического представления данных в векторном формате.

**CSV-файл для табличных данных**

**Формат**: CSV

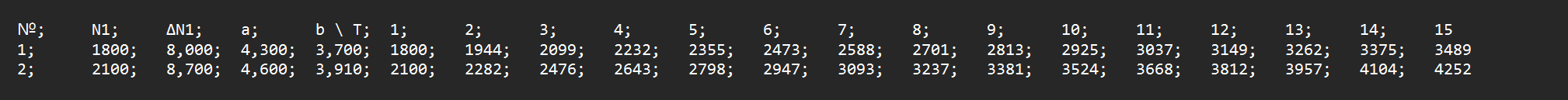
**Обоснование выбора**:

CSV удобен для хранения табличных данных и легко читается большинством программ для обработки данных

CSV-файлы компактны и эффективны для хранения числовых данных в текстовом представлении, что упрощает экспорт и импорт результатов расчетов.

**Символ разделитель:** “ ;\t ”

**Пример CSV-файла**:



**Структура PNG(Portable Network Graphics)-файла**:

Структура PNG-файла задается не вручную, так как это стандартный формат для хранения растрового изображения. Программа будет генерировать график, отображающий изменение интенсивности движения Nt по годам T, и сохранять его как изображение в формате PNG.

График содержит:

Ось X, обозначающую годы эксплуатации (T).

Ось Y, обозначающую интенсивность движения (Nt).

**Пример PNG-файла:**

Изображение выглядит как линия, текст, График, диаграмма

Автоматически созданное описание

**Структура EMF (Enhanced Metafile)-файла:**

**Enhanced Metafile (EMF)** — это векторный формат графического файла, который используется для хранения изображений в виде математически заданных объектов. В отличие от растровых форматов, таких как PNG, EMF сохраняет графику в виде инструкций, позволяя масштабировать изображения без потери качества.

Программа будет генерировать график, отображающий изменение интенсивности движения Nt​ по годам T, и сохранять его как изображение в формате EMF.

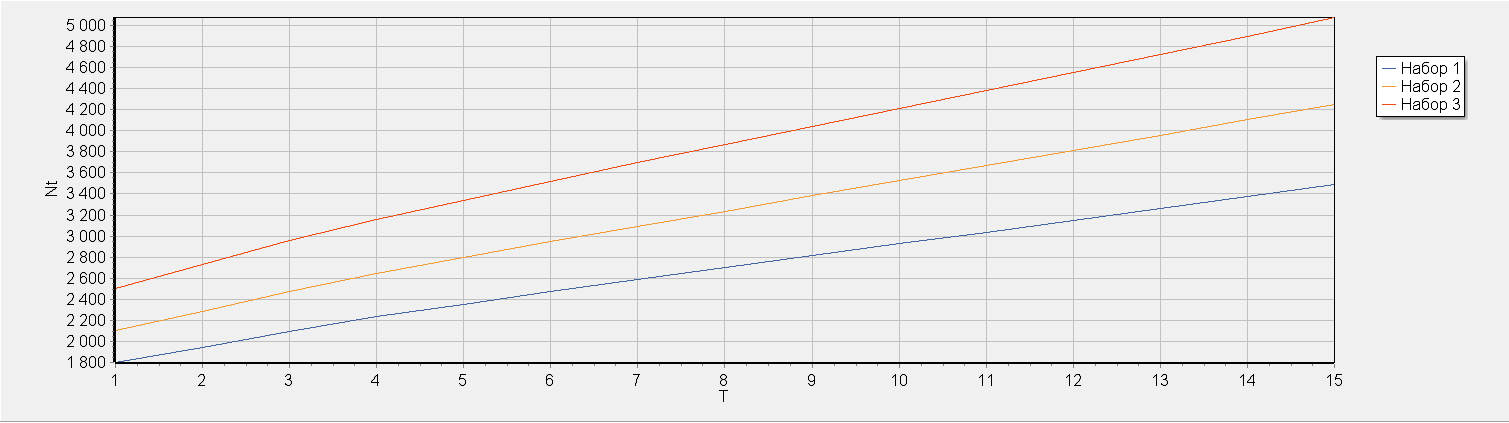
**Преимущества EMF:**

* **Векторная графика:** Все элементы сохраняются в виде векторных объектов, что позволяет масштабировать график без потери качества.
* **Меньший объём памяти:** При сложной графике EMF может занимать меньше места, так как хранит данные в виде инструкций, а не пикселей.
* **Совместимость:** Формат поддерживается Windows и многими программами для редактирования графики.

**График содержит:**

* Ось X, обозначающую годы эксплуатации (T).
* Ось Y, обозначающую интенсивность движения (Nt​).

**Пример EMF-файла:**



# **6. Структура программы**

## **6.1 Модуль <MyUnit>**

### **6.1.3.1 Класс оконной формы <TForm1>**

|  |  |  |
| --- | --- | --- |
| **Имена** | **Класс** | **Назначения** |
| GraphPanel | TPanel | Панель для графика “Интенсивность дорожного движения” |
| Chart1 | TChart | График “Интенсивность дорожного движения” |
| TablePanel | TPanel | Панель для таблицы “Интенсивность дорожного движения” |
| Table | TStringGrid | Таблица “Интенсивность дорожного движения” |
| TableName | TLabel | Подпись к таблице “Интенсивность дорожного движения” |
| Top | TPanel | Панель для хранения двух других панелей |
| TopLeft | TPanel | Панель для таблицы с исходными данными |
| InitialTable | TStringGrid | Таблица с исходными данными |
| CheckBox1 | TCheckBox | Чек-бокс для переключения между режима редактирования |
| TopRight | TPanel | Панель для однострочных редакторов с исходными данными и кнопками |
| EditT0 | TEdit | Однострочный редактор для ввода начального года |
| EditTmax | TEdit | Однострочный редактор для ввода конечного года |
| EditDeltaT | TEdit | Однострочный редактор для ввода интервала между годами |
| LabelT0 | TLabel | Подпись “Начальный год” |
| LabelTmax | TLabel | Подпись “Конечный год” |
| LabelDeltaT | TLabel | Подпись “Интервал между годами” |
| GroupBox | TGroupBox | Группа кнопок для работы с таблицей для исходных данных и вычисления |
| CalcButton | TButton | Кнопка “Вычислить” |
| AddRowButton | TButton | Кнопка “Добавить строку” |
| DelRowButton | TButton | Кнопка “Удалить строку” |
| MainMenu | TMainMenu | Панель меню |
| FileMenu | TMenuItem | Подменю “Файл” |
| Open | TMenuItem | Кнопка “Открыть…” для открытия файла с исходными данными |
| Save | TMenuItem | Кнопка “Сохранить” для сохранения файла с исходными данными |
| SaveAs | TMenuItem | Кнопка “Сохранить как…” для открытия диалога сохранения исходных данных |
| FileCreate | TMenuItem | Кнопка “Создать” для создания пустого файла |
| Exit | TMenuItem | Кнопка “Выход” для выхода из приложения |
| EditMenu | TMenuItem | Подменю “Редактирование” |
| AddRowMenu | TMenuItem | Кнопка “Добавить строку в таблицу с исходными данными” |
| DelRowMenu | TMenuItem | Кнопка “Удалить строку в таблицу с исходными данными” |
| ResultMenu | TMenuItem | Подменю “Результат» |
| CalcMenuButton | TMenuItem | Кнопка “Вычислить” |
| SaveTable | TMenuItem | Кнопка “Сохранить таблицу как…” для открытия диалога сохранения результирующей таблицы |
| SaveGraph | TMenuItem | Кнопка “Сохранить графика как…” для открытия диалога сохранения графика |
| OpenDialog | TOpenDialog | Диалог открытия файла исходных данных |
| SaveDialog | TSaveDialog | Диалог сохранения исходных данных |
| SaveGraphDialog | TSaveDialog | Диалог сохранения результирующей таблицы |
| SaveTableDialog | TSaveDialog | Диалог сохранения графика |
| ImageList | TImageList | Хранение иконок |
| SplitterGT | TSplitter | Разделитель между TablePanel и GraphPanel |
| StatusBar1 | TStatusBar | Строка состояния |

### **6.1.3.2 Методы класса формы**

|  |  |  |
| --- | --- | --- |
| **Имя метода** | **Условие вызова** | **Назначение** |
| OpenClick | Выбор пункта главного меню “Файл”-“Открыть…” | Открытие диалог для загрузки файла с исходными файлами |
| SaveClick | Выбор пункта главного меню “Файл”-“Сохранить” | Сохраняет исходные данные в известный файл или выводит диалог сохранения для выбора файла |
| SaveAsClick | Выбор пункта главного меню “Файл”-“Сохранить как…” | Открывает диалог сохранения |
| FileCreateClick | Выбор пункта главного меню “Файл”-“Создать” | Создание нового пустого файла |
| AddRowMenuClick | Выбор пункта главного меню “Редактирование ”-“Добавить набор в таблицу” или кнопка “Добавить набор” | Добавление в конец таблицы с исходными данными пустой строки |
| DelRowMenuClick | Выбор пункта главного меню “Редактирование ”-“Удалить набор(ы) из таблицы” или кнопка “Удалить набор(ы)” | Удаление выделенных строк из таблицы с исходными данными |
| CalcButtonClick | Выбор пункта главного меню “Результат”-“Вычислить” или кнопка “Вычислить” | Вычисление данных по формуле и вывод из в результирующую таблицу и график |
| SaveTableClick | Выбор пункта главного меню “Результат”-“Сохранить таблицу как… ” | Открывает диалог сохранения результирующей таблицы |
| SaveGraphClick | Выбор пункта главного меню “Результат”-“Сохранить график как… ” | Открывает диалог сохранения графика |
| ValidateKeyPress\_TEdit | Нажатие на клавишу в однострочном редакторе | Предотвращение некорректного ввода в однострочный редактор с клавиатуры |
| ValidateOnChange\_TEdit | Изменение в однострочном редакторе | Предотвращение некорректного ввода в однострочный редактор из буфера обмена |
| ValidateKeyPress\_Table | Нажатие на клавишу в ячейке таблицы | Предотвращение некорректного ввода в ячейку таблицы с клавиатуры |
| ValidateOnChange\_Table | Изменение в ячеке таблицы | Предотвращение некорректного ввода в ячейку таблицу из буфера обмена |
| InitialTableKeyDown | Нажатие на клавишу “Enter” в ячейке таблицы | Выход из режима редактирования |
| InitialTableDblClick | Двойной щелчок по краю заголовка в таблице | Установка оптимальной ширины столбца |
| CheckBox1Click | Щелчок по чек-боксу | Переключени между режимом редактирования и выделения ячеек таблицы |
| TableMouseWheelDown | Прокрутка колёсика мыши вверх | Прокрутка результирующей таблицы влево |
| TableMouseWheelUp | Прокрутка колёсика мыши вниз | Прокрутка результирующей таблицы вправо |
| TableKeyDown | Нажатие кнопки “Shift” | Активация режима пролистывания результирующей таблицы по-вертикали |
| TableDrawCell | Любые изменения в результирующей таблице | Изменение цвета ячеек таблицы |
| FormCloseQuery | Закртие формы или Выбор пункта главного меню “Файл”-“Выход” | Закрытие приложения |

### **6.1.4. Прочие методы**

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя метода** | **Назначение** | **Параметры** | **Возвращаемое значение** |
| bool \_\_fastcall saveData(bool showDialog); | Проверка наличия несохраненных исходных значений и возможное отображение диалога сохранения | showDialog – входной параметр | Возвращает true , если  false – сохранения не было |
| bool \_\_fastcall CheckChangesAndSave() | Проверка на наличие несохранненых изменений, выдача запроса на сохранение, обработка ответов пользователя | - | Возвращает true – изменений не было  false – изменений были, но не сохранены |
| void \_\_fastcall InitialData(const String& fileName) | Запись данных в двоичный файл | fileName – входной параметр | - |
| void \_\_fastcall ValidateInputData(); | Проверка на корректность значений | - | - |
| int \_\_fastcall CalGridColWidth(TStringGrid\* Grid,int Col); | Высчитывает по длине текста ширину столбца | grid –  входной, таблица, в которой нужно высчитать ширину столбца;  int Col – входной, номер столбца | Возвращает ширину столбца |
| void SortGridByColumn(TStringGrid \*grid, int col, bool ascending) | Сортирует строки в таблице | TStringGrid \*grid – входной/выходной, Указатель на объект таблицы  int col – входной, индекс столбца, по которому выполняется сортировка  bool ascending – входной, указывает напрвление сортировки | - |
| void UpdateColumnHeader(TStringGrid \*grid, int col, bool ascending); | Функция обновляет текст заголовка столбца в таблице | TStringGrid \*grid – входной/выходной, Указатель на объект таблицы  int col – входной, индекс столбца, по которому выполняется сортировка  bool ascending – входной, указывает напрвление сортировки | - |

# **7. Описание алгоритмов**

## **7.1 Блок-схема укрупненного алгоритма работы пользовательского интерфейса приложения**

Изображение выглядит как текст, снимок экрана, графический дизайн, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

## **7.2 void \_\_fastcall TForm1::OpenClick(TObject \*Sender)**

Алгоритм обработки выбора пункта меню “Файл” → “Открыть”:

**Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание**

## **7.3 void \_\_fastcall TForm1::SaveClick(TObject \*Sender)**

Алгоритм обработки выбора пункта меню “Файл” → “Сохранить”:

Изображение выглядит как снимок экрана, текст, диаграмма, графический дизайн

Автоматически созданное описание

## **7.4 void \_\_fastcall FormCloseQuery(TObject \*Sender, bool &CanClose)**

Алгоритм обработки выбора пункта меню “Файл” → “Выход” и системного выхода:

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

# **8. Текст программы с комментариями**

## **8.1 Заголовочный файл MyUnit.h**

//---------------------------------------------------------------------------

#ifndef MyUnitH

#define MyUnitH

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

#include <Vcl.ComCtrls.hpp>

#include <Vcl.ToolWin.hpp>

#include <Vcl.Menus.hpp>

#include <Vcl.ExtCtrls.hpp>

#include <Vcl.Grids.hpp>

#include <VCLTee.Chart.hpp>

#include <VclTee.TeeGDIPlus.hpp>

#include <VCLTee.TeEngine.hpp>

#include <VCLTee.TeeProcs.hpp>

#include <VCLTee.Series.hpp>

#include <Vcl.Dialogs.hpp>

#include <System.ImageList.hpp>

#include <Vcl.ImgList.hpp>

#include <Vcl.Imaging.pngimage.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TMainMenu \*MainMenu;

TMenuItem \*FileMenu;

TPanel \*Top;

TPanel \*TablePanel;

TLabel \*TableName;

TPanel \*GraphPanel;

TStringGrid \*Table;

TChart \*Chart1;

TSplitter \*SplitterGT;

TPanel \*TopLeft;

TPanel \*TopRight;

TEdit \*EditT0;

TLabel \*LabelT0;

TLabel \*LabelTmax;

TEdit \*EditTmax;

TLabel \*LabelDeltaT;

TEdit \*EditDeltaT;

TMenuItem \*Open;

TMenuItem \*SaveAs;

TSaveDialog \*SaveDialog;

TOpenDialog \*OpenDialog;

TStatusBar \*StatusBar1;

TMenuItem \*Save;

TStringGrid \*InitialTable;

TMenuItem \*FileCreate;

TMenuItem \*Split;

TMenuItem \*Exit;

TMenuItem \*EditMenu;

TMenuItem \*ResultMenu;

TMenuItem \*AddRowMenu;

TMenuItem \*CalcMenuButton;

TMenuItem \*SaveTable;

TMenuItem \*SaveGraph;

TGroupBox \*GroupBox;

TButton \*CalcButton;

TButton \*AddRowButton;

TButton \*DelRowButton;

TImageList \*ImageList;

TSaveDialog \*SaveTableDialog;

TSaveDialog \*SaveGraphDialog;

TMenuItem \*DelRowMenu;

TCheckBox \*CheckBox1;

TMenuItem \*InsertRowMenu;

TButton \*InsertButton;

void \_\_fastcall CalcButtonClick(TObject \*Sender);

void \_\_fastcall SaveAsClick(TObject \*Sender);

void \_\_fastcall OpenClick(TObject \*Sender);

void \_\_fastcall SaveClick(TObject \*Sender);

void \_\_fastcall ValidateKeyPress\_TEdit(TObject \*Sender, System::WideChar &Key);

void \_\_fastcall ValidateOnChange\_TEdit(TObject \*Sender);

void \_\_fastcall ValidateKeyPress\_Table(TObject \*Sender, System::WideChar &Key);

void \_\_fastcall ValidateOnChange\_Table(TObject \*Sender, System::LongInt ACol, System::LongInt ARow,

const UnicodeString Value);

void \_\_fastcall FileCreateClick(TObject \*Sender);

void \_\_fastcall FormCloseQuery(TObject \*Sender, bool &CanClose);

void \_\_fastcall TableDrawCell(TObject \*Sender, System::LongInt ACol, System::LongInt ARow,

TRect &Rect, TGridDrawState State);

void \_\_fastcall SaveTableClick(TObject \*Sender);

void \_\_fastcall SaveGraphClick(TObject \*Sender);

void \_\_fastcall TableKeyDown(TObject \*Sender, WORD &Key, TShiftState Shift);

void \_\_fastcall TableMouseWheelUp(TObject \*Sender, TShiftState Shift, TPoint &MousePos,

bool &Handled);

void \_\_fastcall TableMouseWheelDown(TObject \*Sender, TShiftState Shift, TPoint &MousePos,

bool &Handled);

void \_\_fastcall CheckBox1Click(TObject \*Sender);

void \_\_fastcall InitialTableKeyDown(TObject \*Sender, WORD &Key, TShiftState Shift);

void \_\_fastcall DelRowMenuClick(TObject \*Sender);

void \_\_fastcall AddRowMenuClick(TObject \*Sender);

void \_\_fastcall InsertRowMenuClick(TObject \*Sender);

void \_\_fastcall InitialTableDblClick(TObject \*Sender);

private: // User declarations

String FileName;

bool \_\_fastcall saveData(bool showDialog);

bool \_\_fastcall CheckChangesAndSave();

void \_\_fastcall InitialData(const String& );

void \_\_fastcall ValidateInputData();

int \_\_fastcall CalGridColWidth(TStringGrid\* Grid,int Col);

void SortGridByColumn(TStringGrid \*grid, int col, bool ascending);

void UpdateColumnHeader(TStringGrid \*grid, int col, bool ascending);

public:

\_\_fastcall TForm1(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

## **8.2 Файл реализации MyUnit.cpp**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include <cstdint>

#include <fstream>

#include <cmath>

#include <set>

#include "MyUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

Caption = L"ТранспортныйПоток - Безымянный";

Table->Cells[0][0] = "№";

Table->ColWidths[0] = 40;

Table->Cells[1][0] = "N1";

Table->Cells[2][0] = "ΔN1";

Table->Cells[3][0] = "a";

Table->Cells[4][0] = "b \\ T";

InitialTable->Cells[0][0] = "№";

InitialTable->Cells[0][1] = "1";

InitialTable->Cells[1][0] = "N1";

InitialTable->Cells[2][0] = "ΔN1";

InitialTable->Cells[3][0] = "a";

InitialTable->Cells[4][0] = "b";

InitialTable->ColWidths[0] = 40;

InitialTable->ColWidths[1] = 130;

InitialTable->ColWidths[2] = 130;

InitialTable->ColWidths[3] = 130;

InitialTable->ColWidths[4] = 130;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::CalcButtonClick(TObject \*Sender)

{

StatusBar1->SimpleText = "";

// Проверка заполненности InitialTable

for (int row = 1; row < InitialTable->RowCount; row++) {

for (int col = 1; col <= 4; col++) {

if (InitialTable->Cells[col][row].Trim().IsEmpty()) {

StatusBar1->SimpleText = "Ошибка: Заполните все данные в строке " + IntToStr(row) + " таблицы.";

return;

}

}

}

try {

ValidateInputData();

} catch (...) {

return;

}

// Проверка заполненности Edit'ов

if (EditT0->Text.Trim().IsEmpty()) {

StatusBar1->SimpleText = L"Ошибка: Поле T0 не заполнено.";

return;

}

if (EditTmax->Text.Trim().IsEmpty()) {

StatusBar1->SimpleText = L"Ошибка: Поле Tmax не заполнено.";

return;

}

if (EditDeltaT->Text.Trim().IsEmpty()) {

StatusBar1->SimpleText = L"Ошибка: Поле ΔT не заполнено.";

return;

}

Table->RowCount = 1;

Table->ColCount = 5;

Chart1->SeriesList->Clear();

// Установка заголовков таблицы

Table->Cells[0][0] = "№";

Table->Cells[1][0] = "N1";

Table->Cells[2][0] = "ΔN1";

Table->Cells[3][0] = "a";

Table->Cells[4][0] = "b \\ T";

uint16\_t T0 = StrToInt(EditT0->Text);

uint16\_t Tmax = StrToInt(EditTmax->Text);

uint8\_t deltaT = StrToInt(EditDeltaT->Text);

int baseRow = 1;

for (int i = 1; i < InitialTable->RowCount; i++) {

try {

int Num = StrToInt(InitialTable->Cells[0][i]);

uint32\_t N1 = StrToInt(InitialTable->Cells[1][i]);

float deltaN1 = StrToFloat(InitialTable->Cells[2][i]);

float a = StrToFloat(InitialTable->Cells[3][i]);

float b = StrToFloat(InitialTable->Cells[4][i]);

// Создание серии для графика

TLineSeries \*NtSeries = new TLineSeries(Chart1);

Chart1->AddSeries(NtSeries);

// Настройка легенды

NtSeries->Title = "Набор " + IntToStr(i);

Chart1->Legend->LegendStyle = lsSeries;

Chart1->Legend->Visible = true;

TColor seriesColor = NtSeries->Color;

if (Table->RowCount <= baseRow)

Table->RowCount = baseRow + 1;

Table->Cells[0][baseRow] = IntToStr(Num);

// Сохраняем цвет в первой колонке для наглядности

Table->Objects[0][baseRow] = reinterpret\_cast<TObject\*>(seriesColor);

// Заполнение исходных данных в таблице

Table->Cells[1][baseRow] = IntToStr(static\_cast<int>(N1));

Table->Cells[2][baseRow] = FormatFloat("0.000", deltaN1);

Table->Cells[3][baseRow] = FormatFloat("0.000", a);

Table->Cells[4][baseRow] = FormatFloat("0.000", b);

// Добавление данных по годам

int colT = 5;

uint32\_t Nt = N1;

float deltaNt;

for (int T = T0; T <= Tmax; T += deltaT) {

if (Table->ColCount <= colT)

Table->ColCount = colT + 1;

Table->Cells[colT][0] = IntToStr(T);

Table->Cells[colT][baseRow] = IntToStr(static\_cast<int>(Nt));

NtSeries->AddXY(T, Nt);

if (T == 1) {

deltaNt = deltaN1;

} else {

deltaNt = (a + b) / std::pow(T - 1, 1.0 / 3.0);

}

Nt = Nt \* (1 + deltaNt / 100.0);

colT++;

}

baseRow++;

} catch (const Exception &e) {

StatusBar1->SimpleText = "Ошибка обработки данных в наборе " + IntToStr(i) + ": " + e.Message;

return;

}

}

StatusBar1->SimpleText = "Расчёт завершён успешно.";

Table->Repaint();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::InitialData(const String& fileName)

{

std::ofstream file(fileName.c\_str(), std::ios::binary);

if (!file)

{

ShowMessage("Ошибка при сохранении файла!");

return;

}

file.exceptions(ios::failbit | ios::badbit);

try

{

uint16\_t T0 = EditT0->Text.IsEmpty() ? 0 : StrToInt(EditT0->Text);

uint16\_t Tmax = EditTmax->Text.IsEmpty() ? 0 : StrToInt(EditTmax->Text);

uint8\_t deltaT = EditDeltaT->Text.IsEmpty() ? 0 : StrToInt(EditDeltaT->Text);

file.write((char\*)&T0, sizeof(T0));

file.write((char\*)&Tmax, sizeof(Tmax));

file.write((char\*)&deltaT, sizeof(deltaT));

uint32\_t rowCount = InitialTable->RowCount - 1; // Исключаем заголовок строки

file.write((char\*)&rowCount, sizeof(rowCount));

for (uint32\_t row = 1; row <= rowCount; row++)

{

uint32\_t N1 = 0;

float deltaN1 = 0.0f;

float a = 0.0f;

float b = 0.0f;

if (!InitialTable->Cells[1][row].Trim().IsEmpty())

N1 = StrToInt(InitialTable->Cells[1][row]); // N1

if (!InitialTable->Cells[2][row].Trim().IsEmpty())

deltaN1 = StrToFloat(InitialTable->Cells[2][row]); // deltaN1

if (!InitialTable->Cells[3][row].Trim().IsEmpty())

a = StrToFloat(InitialTable->Cells[3][row]); // a

if (!InitialTable->Cells[4][row].Trim().IsEmpty())

b = StrToFloat(InitialTable->Cells[4][row]); // b

file.write((char\*)&N1, sizeof(N1));

file.write((char\*)&deltaN1, sizeof(deltaN1));

file.write((char\*)&a, sizeof(a));

file.write((char\*)&b, sizeof(b));

}

file.close();

Caption = "ТранспортныйПоток - " + fileName;

StatusBar1->SimpleText = L"Файл успешно сохранен: " + fileName;

}

catch (const Exception& e)

{

ShowMessage("Ошибка при сохранении данных: " + e.Message);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::SaveAsClick(TObject \*Sender)

{

String fileName;

bool saveCanceled = false;

while (!saveCanceled)

{

if (SaveDialog->Execute())

{

fileName = SaveDialog->FileName;

// Проверка, существует ли файл

if (FileExists(fileName))

{

// Вывод сообщения о перезаписи

if (Application->MessageBox(L"Файл уже существует. Вы хотите перезаписать его?", L"Подтверждение перезаписи", MB\_YESNO | MB\_ICONQUESTION) == IDNO)

{

continue; // Если отмена, возвращаемся к началу цикла

}

}

InitialData(fileName);

Save->Enabled = false;

saveCanceled = true; // Если все в порядке, выходим из цикла

}

else

{

saveCanceled = true; // Если диалог закрыт без сохранения, выходим из цикла

}

}

}

//---------------------------------------------------------------------------

bool \_\_fastcall TForm1::saveData(bool showDialog) {

String svdFileName = SaveDialog->FileName;

if (showDialog) { // Если нужно показать диалог "Сохранить как..."

if (!SaveDialog->Execute()) {

return false;

}

}

// Попытка сохранить файл

try {

InitialData(svdFileName);

Save->Enabled = false;

} catch (Exception &e) {

// Если произошла ошибка сохранения, показать сообщение

Application->MessageBox(

Format(L"Ошибка сохранения файла \"%s\": \"%s\"",

ARRAYOFCONST((SaveDialog->FileName, e.Message))).w\_str(),

Application->Title.w\_str(),

MB\_OK | MB\_ICONERROR);

SaveDialog->FileName = svdFileName; // Вернуть старое имя файла

return false;

}

return true;

}

bool \_\_fastcall TForm1::CheckChangesAndSave() {

if (Save->Enabled) { // Если кнопка "Сохранить" активна, значит есть несохранённые изменения

String fileName;

if (SaveDialog->FileName.IsEmpty()) {

fileName = "Безымянный";

} else {

fileName = SaveDialog->FileName;

}

switch (Application->MessageBox(

(String(L"Сохранить изменения в \"") + fileName + L"\"?").w\_str(),

Application->Title.w\_str(),

MB\_YESNOCANCEL | MB\_ICONQUESTION)) {

case ID\_YES:

if (!saveData(SaveDialog->FileName.IsEmpty())) {

return false;

}

break;

case ID\_CANCEL:

return false;

}

}

return true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::OpenClick(TObject \*Sender)

{

if (!CheckChangesAndSave()) {

return; // Если пользователь решил не сохранять, прерываем действие

}

if (OpenDialog->Execute()) {

for (int row = 1; row < InitialTable->RowCount; row++) {

for (int col = 0; col < InitialTable->ColCount; col++) {

InitialTable->Cells[col+1][row] = "";

}

}

for (int row = 1; row < Table->RowCount; row++) {

for (int col = 0; col < Table->ColCount; col++) {

Table->Cells[col][row] = "";

}

}

Table->RowCount = 1;

Table->ColCount = 5;

// Очистка графика и легенды

Chart1->SeriesList->Clear();

Chart1->Repaint();

FileName = OpenDialog->FileName;

std::ifstream file(FileName.c\_str(), std::ios::binary);

if (!file) {

ShowMessage("Ошибка при открытии файла!");

return;

}

try {

uint16\_t T0, Tmax;

uint8\_t deltaT;

file.read((char\*)&T0, sizeof(T0));

file.read((char\*)&Tmax, sizeof(Tmax));

file.read((char\*)&deltaT, sizeof(deltaT));

EditT0->Text = IntToStr(T0);

EditTmax->Text = IntToStr(Tmax);

EditDeltaT->Text = IntToStr(deltaT);

uint32\_t rowCount;

file.read((char\*)&rowCount, sizeof(rowCount));

InitialTable->RowCount = rowCount + 1;

for (uint32\_t row = 1; row <= rowCount; row++) {

uint32\_t N1;

float deltaN1, a, b;

file.read((char\*)&N1, sizeof(N1));

file.read((char\*)&deltaN1, sizeof(deltaN1));

file.read((char\*)&a, sizeof(a));

file.read((char\*)&b, sizeof(b));

InitialTable->Cells[0][row] = IntToStr(static\_cast<int>(row));

InitialTable->Cells[1][row] = IntToStr(static\_cast<int>(N1));

InitialTable->Cells[2][row] = FormatFloat("0.000", deltaN1);

InitialTable->Cells[3][row] = FormatFloat("0.000", a);

InitialTable->Cells[4][row] = FormatFloat("0.000", b);

}

file.close();

Caption = "ТранспортныйПоток - " + FileName;

StatusBar1->SimpleText = L"Файл успешно открыт: " + FileName;

Save->Enabled = false;

SaveDialog->FileName = OpenDialog->FileName;

InitialTable->Cells[0][0] = "№";

InitialTable->Cells[0][1] = "1";

InitialTable->Cells[1][0] = "N1";

InitialTable->Cells[2][0] = "ΔN1";

InitialTable->Cells[3][0] = "a";

InitialTable->Cells[4][0] = "b";

} catch (const Exception &e) {

file.close();

ShowMessage("Ошибка при чтении данных: " + e.Message);

}

}

if (InitialTable->RowCount > 2){

DelRowButton->Enabled = true;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::ValidateKeyPress\_TEdit(TObject \*Sender, System::WideChar &Key)

{

if (!isdigit(Key) && Key != VK\_BACK && Key != 3 && Key != 22 && Key != 1 && Key != 24 && Key != 26) {

StatusBar1->SimpleText = L"'" + String(Key) + L"' - Недопустимый символ для целого числа!";

Key = 0;

}

if (Sender == EditDeltaT) {

if (Key == '0' && EditDeltaT->Text.IsEmpty()) { // Если вводится 0 и поле будет содержать только 0 запрещаем ввод

StatusBar1->SimpleText = L"Поле ΔT не может быть равно 0!";

Key = 0;

}

}

if (Sender == EditTmax) {

if (Key == '0' && EditDeltaT->Text.IsEmpty()) { // Если вводится 0 и поле будет содержать только 0 запрещаем ввод

StatusBar1->SimpleText = L"Поле Tmax не может быть равно 0!";

Key = 0;

}

}

if (Sender == EditTmax || Sender == EditT0) {

try {

int t0 = EditT0->Text.ToInt();

int tmax = EditTmax->Text.ToInt();

if (Sender == EditTmax) {

String tmaxPreview = EditTmax->Text + Key; // Предполагаемое значение Tmax после ввода

tmax = tmaxPreview.ToInt();

} else if (Sender == EditT0) {

String t0Preview = EditT0->Text + Key; // Предполагаемое значение T0 после ввода

t0 = t0Preview.ToInt();

}

if (tmax <= t0) {

StatusBar1->SimpleText = L"Значение Tmax должно быть больше T0!";

Key = 0;

}

} catch (...) {

// Игнорируем исключения, вызванные временной некорректностью значения

}

}

}

void \_\_fastcall TForm1::ValidateOnChange\_TEdit(TObject \*Sender)

{

bool isValid = true;

if (((TEdit\*)Sender)->Text.IsEmpty()) {

isValid = true;

} else {

try {

StrToUInt(((TEdit\*)Sender)->Text);

}

catch (...) {

isValid = false;

}

}

if (!isValid) {

StatusBar1->SimpleText = L"Некорректное значение!";

} else {

StatusBar1->SimpleText = L"";

}

Save->Enabled = true;

}

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::SaveClick(TObject \*Sender)

{

if (SaveDialog->FileName.IsEmpty())

{

SaveAsClick(Sender);

}

else

{

InitialData(FileName);

Save->Enabled = false;

}

}

//---------------------------------------------------------------------------\*/

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::ValidateKeyPress\_Table(TObject \*Sender, System::WideChar &Key)

{

int col = InitialTable->Col;

switch (col) {

case 1: // N1

if (!isdigit(Key) && Key != VK\_BACK && Key != 3 && Key != 22 && Key != 1 && Key != 24 && Key != 26) {

StatusBar1->SimpleText = L"'" + String(Key) + L"' - Недопустимый символ для целого числа!";

Key = 0;

}

break;

case 2: // deltaN1

case 3: // a

case 4: // b

if (!isdigit(Key) && Key != ',' && Key != VK\_BACK && Key != 3 && Key != 22 && Key != 1 && Key != 24 && Key != 26) {

StatusBar1->SimpleText = L"'" + String(Key) + L"' - Недопустимый символ для числа с плавающей запятой!";

Key = 0;

} else if (Key == ',') {

String currentText = InitialTable->Cells[col][InitialTable->Row];

if (currentText.Pos(',') > 0) {

StatusBar1->SimpleText = L"Повторная запятая недопустима!";

Key = 0;

}

}

break;

default:

Key = 0;

break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::ValidateOnChange\_Table(TObject \*Sender, System::LongInt ACol,

System::LongInt ARow, const UnicodeString Value)

{

bool isValid = true;

try {

switch (ACol) {

case 1: // N1

if (!Value.IsEmpty()) {

StrToUInt(Value);

}

break;

case 2: // deltaN1

case 3: // a

case 4: // b

if (!Value.IsEmpty()) {

StrToFloat(Value);

}

break;

default:

isValid = false;

break;

}

}

catch (...) {

isValid = false;

}

if (!isValid) {

StatusBar1->SimpleText = L"Некорректное значение в таблице!";

} else {

StatusBar1->SimpleText = L"";

}

Save->Enabled = true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FileCreateClick(TObject \*Sender)

{

if (!CheckChangesAndSave()) {

return;

}

SaveDialog->FileName = L"";

EditT0->Text = "";

EditTmax->Text = "";

EditDeltaT->Text = "";

for (int row = 1; row < InitialTable->RowCount; row++) {

for (int col = 0; col < InitialTable->ColCount; col++) {

InitialTable->Cells[col+1][row] = "";

}

}

InitialTable->RowCount = 2;

for (int col = 0; col < InitialTable->ColCount; col++) {

InitialTable->Cells[col+1][1] = "";

}

for (int row = 1; row < Table->RowCount; row++) {

for (int col = 0; col < Table->ColCount; col++) {

Table->Cells[col][row] = "";

}

}

Table->RowCount = 1;

Table->ColCount = 5;

// Очистка графика

for (int i = 0; i < Chart1->SeriesCount(); i++) {

Chart1->Series[i]->Clear();

}

Chart1->Legend->Visible = false;

Save->Enabled = false;

DelRowButton->Enabled = false;

for (int i = 1; i < InitialTable->RowCount; ++i) {

InitialTable->Cells[0][i] = IntToStr(i);

}

InitialTable->FixedRows=1;

InitialTable->Cells[0][0] = "№";

InitialTable->Cells[0][1] = "1";

InitialTable->Cells[1][0] = "N1";

InitialTable->Cells[2][0] = "ΔN1";

InitialTable->Cells[3][0] = "a";

InitialTable->Cells[4][0] = "b";

Caption = "ТранспортныйПоток - Безымянный";

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormCloseQuery(TObject \*Sender, bool &CanClose)

{

if (!CheckChangesAndSave())

CanClose = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::TableDrawCell(TObject \*Sender, System::LongInt ACol, System::LongInt ARow,

TRect &Rect, TGridDrawState State)

{

TStringGrid \*grid = static\_cast<TStringGrid\*>(Sender);

if (ARow == 0 && ACol == 4) // Заголовок столбца "b \\ T"

{

grid->Canvas->Brush->Color = clSilver;

grid->Canvas->FillRect(Rect);

grid->Canvas->Font->Color = clBlack;

String textLeft = "b";

String textMiddle = "\\";

String textRight = "T";

int leftTextWidth = grid->Canvas->TextWidth(textLeft);

int middleTextWidth = grid->Canvas->TextWidth(textMiddle);

int rightTextWidth = grid->Canvas->TextWidth(textRight);

int leftX = Rect.Left + 2;

int middleX = Rect.Left + (Rect.Width() - middleTextWidth) / 2;

int rightX = Rect.Right - rightTextWidth - 6;

grid->Canvas->TextOut(leftX, Rect.Top + 2, textLeft);

grid->Canvas->TextOut(middleX, Rect.Top + 2, textMiddle);

grid->Canvas->TextOut(rightX, Rect.Top + 2, textRight);

}

else if (ACol == 0 && ARow == 0){

grid->Canvas->Brush->Color = clGray;

grid->Canvas->FillRect(Rect);

grid->Canvas->Font->Color = clBlack;

grid->Canvas->TextOut(Rect.Left + 2, Rect.Top + 2, grid->Cells[ACol][ARow]);

}

else if (ACol == 0 && ARow > 0)

{

TColor cellColor = static\_cast<TColor>(reinterpret\_cast<intptr\_t>(grid->Objects[ACol][ARow]));

grid->Canvas->Brush->Color = cellColor;

grid->Canvas->FillRect(Rect);

grid->Canvas->Font->Color = clBlack;

grid->Canvas->TextOut(Rect.Left + 2, Rect.Top + 2, grid->Cells[ACol][ARow]);

}

else if (ACol >= 1 && ACol <= 4 && ARow > 0)

{

grid->Canvas->Brush->Color = clWebLightgrey;

grid->Canvas->FillRect(Rect);

grid->Canvas->Font->Color = clBlack;

grid->Canvas->TextOut(Rect.Left + 2, Rect.Top + 2, grid->Cells[ACol][ARow]);

}

else if (ACol >= 1 && ACol <= 4 && ARow == 0){

grid->Canvas->Brush->Color = clSilver;

grid->Canvas->FillRect(Rect);

grid->Canvas->Font->Color = clBlack;

grid->Canvas->TextOut(Rect.Left + 2, Rect.Top + 2, grid->Cells[ACol][ARow]);

}

if (ACol == 4)

{

grid->Canvas->Pen->Width = 2;

grid->Canvas->Pen->Color = clBlack;

grid->Canvas->MoveTo(Rect.Right - 1, Rect.Top);

grid->Canvas->LineTo(Rect.Right - 1, Rect.Bottom);

}

if (ACol > 4)

{

grid->DefaultDrawing = true;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::SaveTableClick(TObject \*Sender)

{

String fileName;

bool saveCanceled = false;

while(!saveCanceled)

{

if (SaveTableDialog->Execute())

{

fileName = SaveTableDialog->FileName;

if (FileExists(fileName))

{

if (Application->MessageBox(L"Файл уже существует. Вы хотите перезаписать его?", L"Подтверждение перезаписи", MB\_YESNO | MB\_ICONQUESTION) == IDNO)

{

continue;

}

}

std::ofstream file;

try

{

file.open(fileName.c\_str(), std::ios::out | std::ios::trunc);

if (!file.is\_open())

{

ShowMessage(L"Ошибка: Не удалось открыть файл для записи.");

break;

}

for (int row = 0; row < Table->RowCount; row++)

{

for (int col = 0; col < Table->ColCount; col++)

{

String cellData = Table->Cells[col][row];

if (cellData.IsEmpty())

{

cellData = "";

}

file << AnsiString(cellData).c\_str();

if (col < Table->ColCount - 1)

{

file << ";\t"; // Разделитель для Excel

}

}

file << "\n";

}

file.close();

StatusBar1->SimpleText = L"Таблица успешно сохранена в файл: " + fileName;

}

catch (const std::exception &e)

{

if(file.is\_open())

file.close();

ShowMessage(L"Ошибка при сохранении файла: " + String(e.what()));

}

saveCanceled = true;

}

else {

saveCanceled = true;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::SaveGraphClick(TObject \*Sender)

{

String fileName;

bool saveCanceled = false;

while (!saveCanceled)

{

if (SaveGraphDialog->Execute())

{

fileName = SaveGraphDialog->FileName;

int selectedFilterIndex = SaveGraphDialog->FilterIndex;

String fileExtension;

if (FileExists(fileName))

{

if (Application->MessageBox(L"Файл уже существует. Вы хотите перезаписать его?", L"Подтверждение перезаписи", MB\_YESNO | MB\_ICONQUESTION) == IDNO)

{

continue;

}

}

if (selectedFilterIndex == 1) // Сохранение в PNG

{

fileExtension = ".png";

TBitmap \*bitmap = new TBitmap();

try

{

Chart1->SaveToBitmapFile(L"temp\_graph.bmp");

bitmap->LoadFromFile(L"temp\_graph.bmp");

TPngImage \*png = new TPngImage();

try

{

png->Assign(bitmap);

// Добавляем расширение, если его нет

if (!SameText(ExtractFileExt(fileName), fileExtension))

{

fileName = ChangeFileExt(fileName, fileExtension);

}

png->SaveToFile(fileName); // Сохранить в PNG

}

\_\_finally

{

delete png;

}

DeleteFile(L"temp\_graph.bmp");

StatusBar1->SimpleText = L"График успешно сохранен в файл: " + fileName;

}

\_\_finally

{

delete bitmap;

}

}

else if (selectedFilterIndex == 2) // Сохранение в EMF

{

fileExtension = ".emf";

try

{

// Добавляем расширение, если его нет

if (!SameText(ExtractFileExt(fileName), fileExtension))

{

fileName = ChangeFileExt(fileName, fileExtension);

}

TMetafile \*metafile = new TMetafile();

try

{

metafile->Enhanced = true;

TMetafileCanvas \*canvas = new TMetafileCanvas(metafile, 0);

try

{

Chart1->Draw(canvas, Rect(0, 0, Chart1->Width, Chart1->Height));

}

\_\_finally

{

delete canvas;

}

metafile->SaveToFile(fileName);

StatusBar1->SimpleText = L"График успешно сохранен в файл: " + fileName;

}

\_\_finally

{

delete metafile;

}

}

catch (...)

{

Application->MessageBox(L"Ошибка при сохранении графика в формате EMF.", L"Ошибка", MB\_ICONERROR);

}

}

else

{

Application->MessageBox(L"Неверный формат файла.", L"Ошибка формата", MB\_ICONERROR);

}

saveCanceled = true;

}

else

{

saveCanceled = true;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::TableKeyDown(TObject \*Sender, WORD &Key, TShiftState Shift)

{

if (Shift.Contains(ssShift)) // Проверяем, зажат ли Shift

{

switch (Key)

{

case VK\_LEFT: // Стрелка влево

Table->LeftCol = std::max(0, Table->LeftCol - 1);

break;

case VK\_RIGHT: // Стрелка вправо

Table->LeftCol = std::min(Table->ColCount - 1, Table->LeftCol + 1);

break;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::TableMouseWheelUp(TObject \*Sender, TShiftState Shift, TPoint &MousePos,

bool &Handled)

{

if (Shift.Contains(ssShift)) // Проверяем, зажат ли Shift

{

Table->LeftCol = std::max(0, Table->LeftCol - 1); // Прокрутка влево

Handled = true;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::TableMouseWheelDown(TObject \*Sender, TShiftState Shift, TPoint &MousePos,

bool &Handled)

{

if (Shift.Contains(ssShift)) // Проверяем, зажат ли Shift

{

Table->LeftCol = std::min(Table->ColCount - 1, Table->LeftCol + 1); // Прокрутка вправо

Handled = true;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::ValidateInputData()

{

for (int row = 1; row < InitialTable->RowCount; row++)

{

try

{

StrToInt(InitialTable->Cells[1][row]); // N1

StrToFloat(InitialTable->Cells[2][row]); // deltaN1

StrToFloat(InitialTable->Cells[3][row]); // a

StrToFloat(InitialTable->Cells[4][row]); // b

}

catch (const Exception &e)

{

StatusBar1->SimpleText = "Ошибка: Некорректные данные в строке " + IntToStr(row);

throw;

}

}

try

{

StrToInt(EditT0->Text);

}

catch (const Exception &e)

{

StatusBar1->SimpleText = "Ошибка: Некорректные данные в поле T0.";

throw;

}

try

{

StrToInt(EditTmax->Text);

}

catch (const Exception &e)

{

StatusBar1->SimpleText = "Ошибка: Некорректные данные в поле Tmax.";

throw;

}

try

{

StrToInt(EditDeltaT->Text);

}

catch (const Exception &e)

{

StatusBar1->SimpleText = "Ошибка: Некорректные данные в поле ΔT.";

throw;

}

if(StrToInt(EditDeltaT->Text) == 0){

StatusBar1->SimpleText = L"Поле ΔT не может быть равно 0!";

throw;

}

}

void \_\_fastcall TForm1::CheckBox1Click(TObject \*Sender)

{

if (CheckBox1->Checked) {

InitialTable->Options = InitialTable->Options << goEditing;

} else {

InitialTable->Options = InitialTable->Options >> goEditing;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::InitialTableKeyDown(TObject \*Sender, WORD &Key, TShiftState Shift)

{

static String oldValue;

if (Key == VK\_RETURN) {

Key = 0;

if (InitialTable->EditorMode) {

oldValue = InitialTable->Cells[InitialTable->Col][InitialTable->Row];

}

InitialTable->EditorMode = !InitialTable->EditorMode;

if (InitialTable->EditorMode) {

oldValue = InitialTable->Cells[InitialTable->Col][InitialTable->Row];

}

}

if (Key == VK\_ESCAPE) {

Key = 0;

if (InitialTable->EditorMode) {

InitialTable->Cells[InitialTable->Col][InitialTable->Row] = oldValue;

InitialTable->EditorMode = false;

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::DelRowMenuClick(TObject \*Sender)

{

int topRow = InitialTable->Selection.Top;

int bottomRow = InitialTable->Selection.Bottom;

if (topRow == -1 || bottomRow == -1)

return;

String message = L"Вы уверены, что хотите удалить строки: ";

for (int i = topRow; i <= bottomRow; ++i) {

if (i < InitialTable->RowCount) {

message += IntToStr(i) + L", ";

}

}

if (message.Length() > 2) {

message = message.SubString(1, message.Length() - 2);

}

message += L"?";

int result = Application->MessageBox(

message.w\_str(),

Application->Title.w\_str(),

MB\_YESNO | MB\_ICONQUESTION | MB\_DEFBUTTON2); // кнопка "Нет" по умолчанию

if (result == IDYES) {

for (int i = bottomRow; i >= topRow; --i) {

if (i < InitialTable->RowCount) {

// Сдвигаем строки вверх

for (int j = i; j < InitialTable->RowCount - 1; ++j) {

for (int col = 0; col < InitialTable->ColCount; ++col) {

InitialTable->Cells[col][j] = InitialTable->Cells[col][j + 1];

}

}

// Очищаем последнюю строку после сдвига

if (InitialTable->RowCount > 0) {

for (int col = 0; col < InitialTable->ColCount; ++col) {

InitialTable->Cells[col][InitialTable->RowCount - 1] = "";

}

}

// Уменьшаем количество строк

InitialTable->RowCount--;

}

}

// Перенумеровываем первую колонку

for (int i = 1; i < InitialTable->RowCount; i++) {

InitialTable->Cells[0][i] = IntToStr(i);

}

}

if (InitialTable->RowCount <= 1) {

DelRowMenu->Enabled = false;

DelRowButton->Enabled = false;

StatusBar1->SimpleText = L"В таблице должна быть хотя бы одна фиксированная строка.";

CheckBox1->Enabled = false;

CheckBox1->Checked = false;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::AddRowMenuClick(TObject \*Sender)

{

int newRow = InitialTable->RowCount;

InitialTable->RowCount++;

InitialTable->Cells[0][newRow] = IntToStr(newRow);

if (InitialTable->RowCount >= 1) {

DelRowMenu->Enabled = true;

DelRowButton->Enabled = true;

CheckBox1->Enabled = true;

CheckBox1->Checked = true;

InitialTable->FixedRows = 1;

}

InitialTable->Row = newRow;

}

//---------------------------------------------------------------------------

int \_\_fastcall TForm1::CalGridColWidth(TStringGrid\* Grid, int Col) {

TCanvas\* Canvas = new TCanvas();

try {

Canvas->Handle = Grid->Canvas->Handle;

Canvas->Font = Grid->Font;

int maxWidth = 0;

for (int row = 0; row < Grid->RowCount; ++row) {

TRect textRect;

DrawTextW(Canvas->Handle, Grid->Cells[Col][row].c\_str(), -1, &textRect, DT\_CALCRECT | DT\_SINGLELINE);

maxWidth = std::max(maxWidth, textRect.Width());

}

return maxWidth + 10;

}

\_\_finally {

delete Canvas;

}

}

void \_\_fastcall TForm1::InsertRowMenuClick(TObject \*Sender)

{

int currentRow = InitialTable->Row;

if (currentRow < 0) return;

int newRow = currentRow + 1;

InitialTable->RowCount++;

for (int i = InitialTable->RowCount - 2; i >= newRow; i--) {

for (int j = 0; j < InitialTable->ColCount; j++) {

InitialTable->Cells[j][i + 1] = InitialTable->Cells[j][i];

}

}

for (int j = 0; j < InitialTable->ColCount; j++) {

InitialTable->Cells[j][newRow] = "";

}

InitialTable->Cells[0][newRow] = IntToStr(newRow + 1);

for (int i = 1; i < InitialTable->RowCount; i++) {

InitialTable->Cells[0][i] = IntToStr(i);

}

InitialTable->Row = newRow;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::InitialTableDblClick(TObject \*Sender)

{

TPoint p;

GetCursorPos(&p);

TStringGrid\* grid = dynamic\_cast<TStringGrid\*>(Sender);

if (grid) {

p = grid->ScreenToClient(p);

if (GetCursor() == Screen->Cursors[crHSplit]) { // курсор на краю?

TGridCoord gc = grid->MouseCoord(p.X - 4, p.Y);

int width = CalGridColWidth(grid, gc.X);

if (width < 5) {

width = 5;

}

grid->ColWidths[gc.X] = width;

}

else {

TGridCoord gc = grid->MouseCoord(p.X, p.Y); // иначе по заголовку

if (gc.Y == 0) {

int col = gc.X;

static int lastSortedCol = -1; // если уже отсортирован

static bool ascending = true;

if (lastSortedCol == col) { // смена напрвления

ascending = !ascending;

} else {

ascending = true;

}

if (lastSortedCol != -1 && lastSortedCol != col) { // удаляем стрелочку с пердыдущего

String prevHeaderText = grid->Cells[lastSortedCol][0];

int arrowPos = prevHeaderText.Pos("▲");

if (arrowPos > 0) {

prevHeaderText = prevHeaderText.SubString(1, arrowPos - 1);

}

else {

arrowPos = prevHeaderText.Pos("▼");

if (arrowPos > 0) {

prevHeaderText = prevHeaderText.SubString(1, arrowPos - 1);

}

}

grid->Cells[lastSortedCol][0] = prevHeaderText;

}

lastSortedCol = col;

SortGridByColumn(grid, col, ascending);

String arrow = ascending ? "▲" : "▼";

String headerText = grid->Cells[col][0];

int arrowPos = headerText.Pos("▲");

if (arrowPos > 0) {

headerText = headerText.SubString(1, arrowPos - 1);

}

else {

arrowPos = headerText.Pos("▼");

if (arrowPos > 0) {

headerText = headerText.SubString(1, arrowPos - 1);

}

}

headerText = headerText + arrow;

grid->Cells[col][0] = headerText;

}

}

}

}

void TForm1::SortGridByColumn(TStringGrid \*grid, int col, bool ascending)

{

int rowCount = grid->RowCount;

bool swapped;

do {

swapped = false;

for (int i = 1; i < rowCount - 1; i++) {

String cell1 = grid->Cells[col][i];

String cell2 = grid->Cells[col][i + 1];

bool condition = (ascending ? cell1 > cell2 : cell1 < cell2);

if (condition) {

for (int j = 1; j < grid->ColCount; j++) {

String temp = grid->Cells[j][i]; // строки меняем местами

grid->Cells[j][i] = grid->Cells[j][i + 1];

grid->Cells[j][i + 1] = temp;

}

swapped = true;

}

}

} while (swapped);

}