|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Министерство науки и высшего образования Российской Федерации | | | | | | | | | | |
| Федеральное государственное бюджетное образовательное учреждение | | | | | | | | | | |
| высшего образования | | | | | | | | | | |
| **«Тверской государственный технический университет»** | | | | | | | | | | |
| **(ТвГТУ)** | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| Кафедра Программного обеспечения | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| **Лабораторная работа № 1** | | | | | | | | | | |
| по дисциплине | | | | | | | | | | |
| **«Объектно-Ориентированное Программирование»** | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Выполнил: | | | | | | | **Лёвкин Д.А** | | | |
| Группа: | | | | | | | **Б.ПИН.ИИ.24.16** | | | |
|  | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Тверь, 2025 | | | | | | | | | | |

1. Трудности, с которыми столкнулись при разработке приложения

При разработке приложения «Мастер работы с массивами» были выявлены следующие трудности:

1. Неправильная структура интерфейса в XAML  
   На начальном этапе вкладки (TabItem) не были обёрнуты в TabControl, что приводило к ошибкам компиляции и неработоспособности интерфейса. Это было связано с недостаточным пониманием иерархии элементов управления в WPF. После изучения документации и исправления структуры проблема была устранена.
2. Динамическая работа с типами данных  
   Реализация ввода элементов массива с учётом выбранного типа данных (int, float, DateTime) оказалась сложной. Сначала все поля ввода были текстовыми, что неудобно для дат. Однако после анализа требований стало ясно, что на первой вкладке нужно только выбрать тип, а уже на второй — использовать соответствующие контролы. В финальной версии ввод реализован через TextBox с проверкой формата, что соответствует заданию.
3. Блокировка/разблокировка вкладок и элементов  
   Логика блокировки первой вкладки после добавления первого элемента потребовала тщательной проработки. Необходимо было отслеживать изменение каждого поля ввода и определять момент, когда первый элемент становится не-null. Это было реализовано через флаг \_isFirstElementAdded и проверку массива при каждом изменении.
4. Работа с деревом файлов (TreeView)  
   Начальные попытки заполнить TreeView без использования модели данных (FileNode) оказались неудачными. Пришлось создать отдельный класс для представления узлов дерева и правильно организовать привязку данных (HierarchicalDataTemplate), чтобы отображалась иерархия.

## **2. Обработка ошибок**

Обработка исключений реализована во всех ключевых методах приложения. Используются блоки try-catch для перехвата и корректного отображения ошибок пользователю.

### **Примеры обработки ошибок:**

#### 1. **Ввод элемента массива**

private void OnArrayElementTextChanged(object sender, TextChangedEventArgs e)

{

try

{

// ... парсинг и валидация значения

}

catch (FormatException)

{

ShowErrorMessage("Неверный формат данных. Проверьте ввод.");

}

catch (Exception ex)

{

ShowErrorMessage($"Ошибка обработки ввода: {ex.Message}");

}

}

Если пользователь введёт "abc" вместо числа или даты — программа не сломается, а покажет информативное сообщение.

#### 2. **Сохранение массива в файл**

private void OnMenuSaveClick(object sender, RoutedEventArgs e)

{

try

{

var dialog = new SaveFileDialog();

if (dialog.ShowDialog() == true)

{

SaveArrayToFile(dialog.FileName);

AddFileToTree(dialog.FileName);

}

}

catch (UnauthorizedAccessException)

{

ShowErrorMessage("Нет прав для сохранения файла в выбранной директории");

}

catch (IOException ioEx)

{

ShowErrorMessage($"Ошибка ввода-вывода: {ioEx.Message}");

}

catch (Exception ex)

{

ShowErrorMessage($"Неизвестная ошибка: {ex.Message}");

}

}

Здесь обрабатываются:

* Отсутствие прав доступа
* Ошибки ввода-вывода (например, диск извлечён)
* Любые другие непредвиденные ошибки

#### 3. **Загрузка содержимого файла**

private void OnLoadFileContentClick(object sender, RoutedEventArgs e)

{

try

{

var selectedNode = filesTreeView.SelectedItem as FileNode;

if (selectedNode == null || selectedNode.IsDirectory) return;

LoadFileContent(selectedNode.Name);

}

catch (Exception ex)

{

ShowErrorMessage($"Ошибка при загрузке содержимого: {ex.Message}");

}

}

### **Общие принципы обработки ошибок:**

* Все исключения перехватываются.
* Пользователю показывается понятное сообщение через MessageBox.
* Исходные данные не теряются.
* Приложение продолжает работать дальше.

## **Вывод**

В ходе выполнения лабораторной работы были изучены основные элементы WPF для создания графического интерфейса. Реализовано приложение с несколькими вкладками, поддерживающее работу с массивами различных типов, их сортировку и управление файлами.

Удалось:

* Освоить компоновку интерфейса с помощью Grid, DockPanel, StackPanel.
* Научиться работать с событиями и привязкой данных.
* Реализовать обработку исключений и защиту от некорректного ввода.
* Создать удобный и понятный интерфейс, соответствующий требованиям.

Приложение готово к демонстрации и дальнейшему развитию (например, добавление MVVM, поддержка больше типов, экспорт в Excel и т.д.).

**Скриншоты работы приложения:**

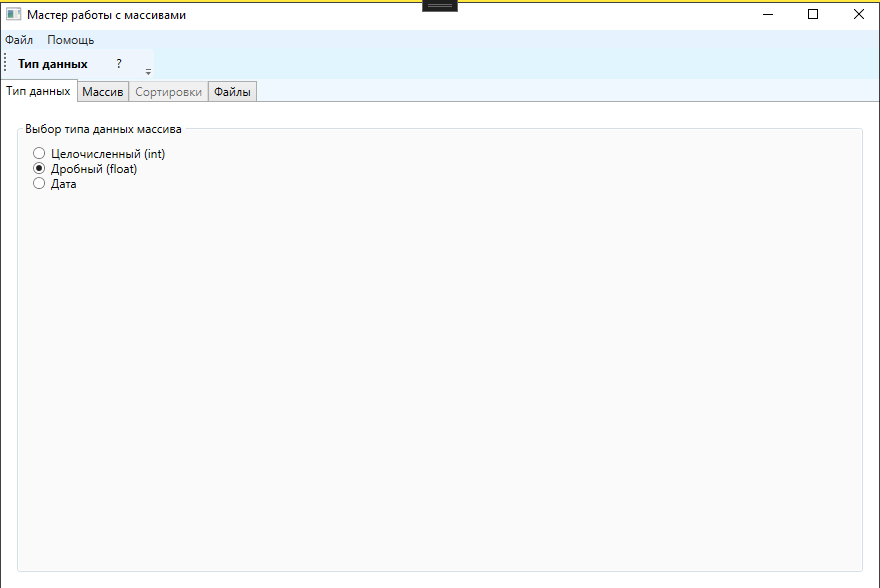
****

Рис. 1 – вкладка «Типы данных»

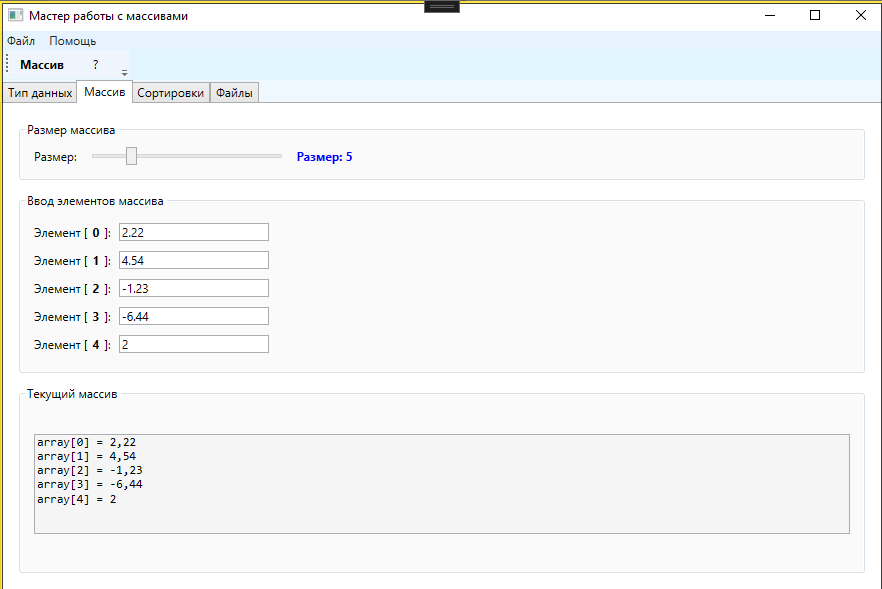
****

Рис. 2 – вкладка «Массив»

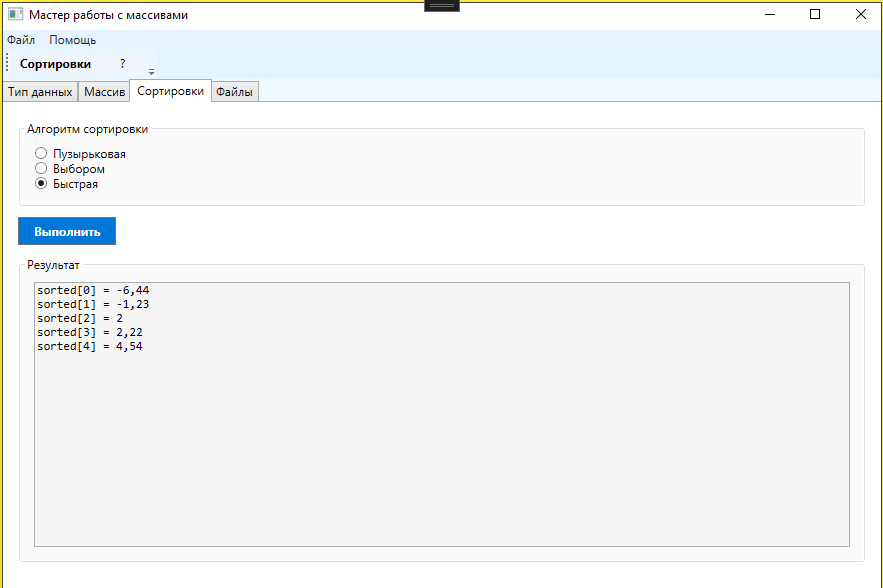
****

Рис. 3 – вкладка «Сортировки»