Лабораторная работа № 4. Обработчики ошибок. Использование

модулей и библиотек. Публикация проектов

**Задание 4-1**. Создать процесс автоматизации, который построит таблицу данных, и заполнит этими данными файл **.csv**. Для проверки введенных данных использовать механизм **Try-Catch**.

1. Создать процесс и назвать его **SimpleTryCatch**.

2. Перенести действие **Sequence** на рабочую панель.

3. Добавить действие **Try Catch.**

4. Добавить действие **Sequence** внутрь блока **Try** и назвать **‘Получение входных данных и ввод их в таблицу’**.

Создать переменные **Name** и **Age** типа **String**, переменную **DtTable** типа **DataTable**.

5. Добавить действие **Input Dialog**.

Указать в свойствах в поле **Title** значение **"Имя"**, в поле **Label** текст **"Введите имя**", в поле **Результат** переменную **Name**.

6. Добавить другое действие **Input Dialog**.

Указать в свойстве **Title** значение "**Возраст**", в свойстве **Label** текст **"Введите возраст в правильном или неправильном формате"**, в поле **Результат** переменную **Age**.

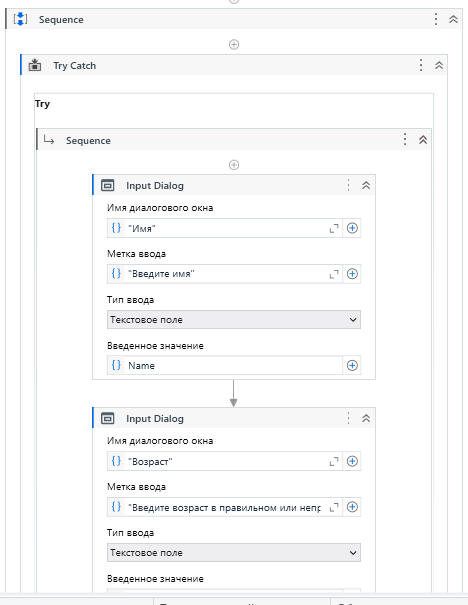


Рисунок 4.1 – Блок Try

7. Добавить действие **Build Data Table.**

В свойствах указать переменную **DtTable** в поле **Результат**.

Сделать двойной щелчок по кнопке **DataTable**. В открывшемся окне дать столбцам названия **Name** (тип **String**)и **Age** (тип **Int32**).

8. Добавить действие **Add Data Row**.

В свойстве **ArrayRow** указать массив переменных, из которых будут взяты данные для заполнения таблицы: **{Name,Age}**.

Указать переменную **DtTable** в поле **DataTable**.

9. Добавить действие **Write CSV**.

Объявить переменную **DtTable** в поле **DataTable** и ввести название файла **"Untitled.csv"**, в который будут сохранены данные, в свойство **FilePath**.

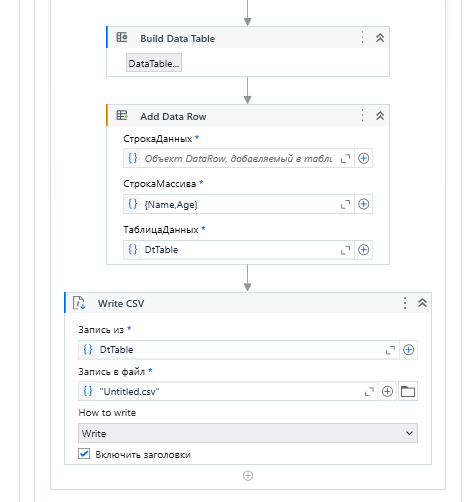


Рисунок 4.2 – Блок Try

10. Щелкнуть в блоке **Catches** и в раскрывающемся списке поля исключения выбрать **System.Exception**.

11. Добавить действие **Message Box** в созданную последовательность **Sequence.**

В свойствах в поле **Text** написать **"Исключение: добавлены недопустимые данные".**

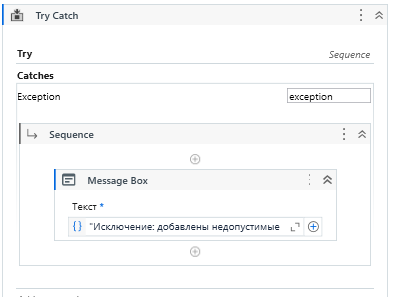
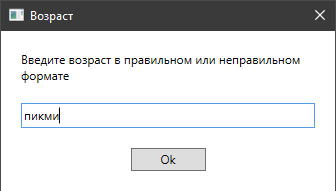
****

Рисунок 4.3 – Блок Cathes

12.Запустить процесс автоматизации на выполнение. Опробовать выполнение проекта при вводе данных в правильном и неправильном формате.

****

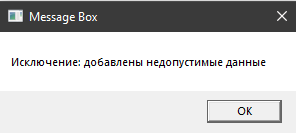


Рисунок 4.4 – Результат работы

**Задание 4-2**. Создать процесс автоматизации с использованием Глобального обработчика исключений, который после ввода текста в Блокнот закроет приложение.

Если документ сохранить нажатием клавиш **Ctrl-s** после автоматического ввода текста, то процесс заканчивается правильно. Если не сохранять документ, то появится сообщение об ошибке.

1. Создать **процесс и назвать его GlobalException**.

2. Открыть Блокнот и сохранить файл.

3. Добавить действие **[Type Into](https://activities.uipath.com/docs/type-into" \t "_self)** на рабочую панель. Действие будет автоматически добавлено в последовательность **Sequence**.

Ввести текст, например, “**Глобальный обработчик исключений**” в поле **Text** (этот текст будет записан в Блокнот).

Нажать ссылку **Указать на экране**. Окно Studio сворачивается и позволяет щелкнуть внутри окна Блокнота.

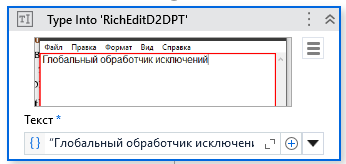


Рисунок 4.5 – действие Type Into

4. Добавить действие **Send Hotkey** в рабочий процесс.

Отметить **Ctrl**, ввести в поле **Key** значение **s** (для сохранения документа Блокнота после ввода текста). Указать на окно Блокнота.

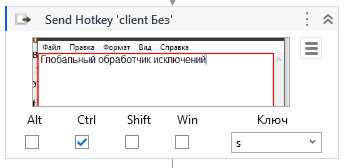


Рисунок 4.6 – действие Send Hotkey

5. Добавить в рабочий процесс действие **Close Application**.

Указать на окно Блокнота.

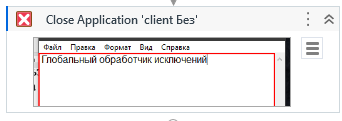


Рисунок 4.7 – действие Close Application

**Добавление в проект глобального обработчика исключений**

6. На **ленте** **Проект** **нажать кнопку Новая и выбрать Глобальный обработчик**.

Откроется окно, в котором надо ввести **Имя** обработчика и сохранить его в папке проекта.

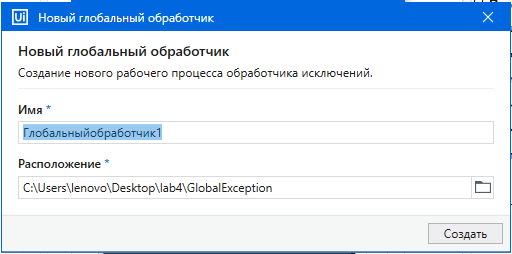


Рисунок 4.8 – добавление Глобального обработчика

7. Запустить процесс автоматизации **Main** на выполнение.

После записи текста в Блокнот нажать вместе клавиши **Ctrl-s**, чтобы правильно закрыть и сохранить файл.

8. Вновь запустить процесс автоматизации на выполнение. После записи текста в Блокнот не нажимать клавиши **Ctrl-s**. После некоторого времени появится сообщение об ошибке.

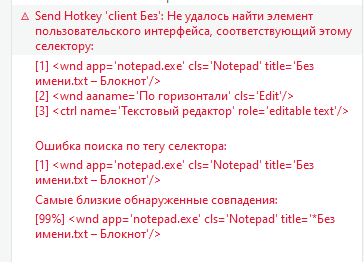


Рисунок 4.9 – сообщение об ошибке

9. Открыть **Глобальный обработчик** и ознакомиться с ним.

Условие **errorInfo.RetryCount < 3** определяет количество исключений.

В переменной **result** метод **ErrorAction**.**Retry** осуществляет попытку выполнения действия, вызвавшего исключение.

Если исключение появилось три раза, то метод **ErrorAction.Abort** прерывает выполнение процесса и выдается соответствующее сообщение.

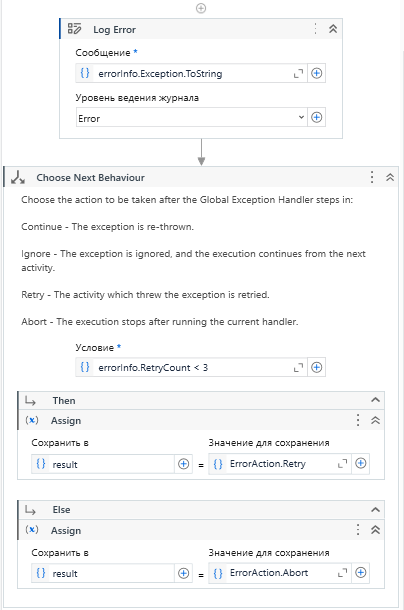
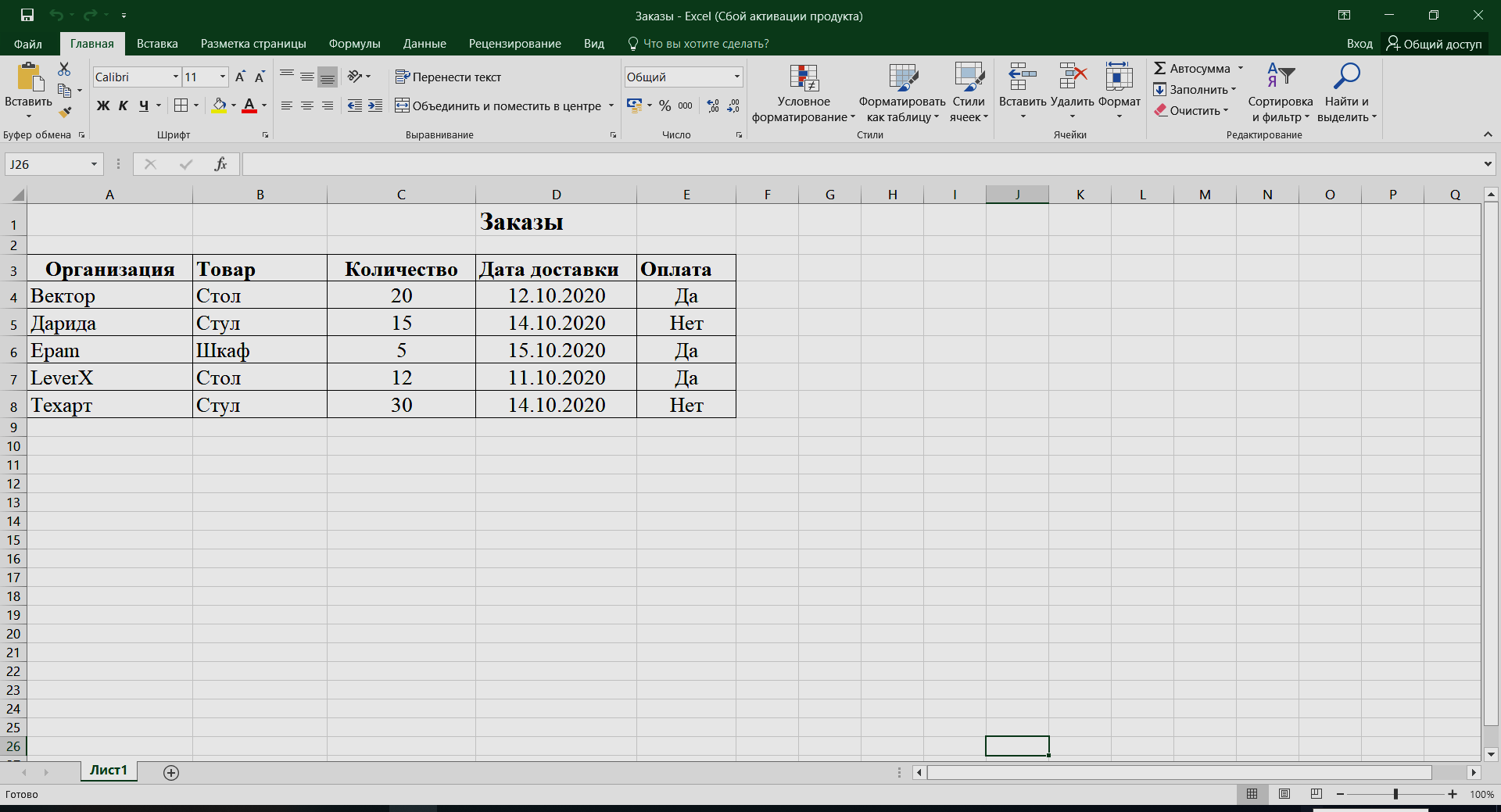


Рисунок 4.10 – Глобальный обработчик

**Задание 4-3**. Создать процесс автоматизации, демонстрирующий способы использования **модуля** в процессе автоматизации с действием **Invoke Workflow File**. Процесс автоматизации должен анализировать содержимое таблицы Excel, в которой записана информация о заказанных организациями товарах и оплате за них. Если товар не оплачен, то название организации следует вывести в окно **Вывод**.



Пусть имеется таблица Excel c информацией о заказанных организациями товарах с именем **Заказы.xlsx**.

В **модуле** сравнивается содержимое столбца с признаком оплаты со словом "Нет". Если товар не оплачен, то в окно **Вывод** выводится соответствующее название организации.

В основной программе читается таблица и для каждой строки осуществляется обращение к модулю.

1. Создать новый процесс и назвать его **OrderWithModule.**

2. Перенести последовательность **Sequence** на рабочую панель.

3. Добавить действие **Excel Application Scope**.

Ввести путь к таблице Excel "**Заказы.xlsx**", если таблица лежит в папке **OrderModule** или ввести полный путь к файлу, если таблица в другой папке, в свойство **Workbook path**.

5. Добавить действие **Read Range**.

В поле **SheetName** ввести текст **"Лист1"**, в поле **Range** ввести **"A3:E8"**, в поле **DataTable** создать переменную **DTable**, используя клавиши **Ctrl-K**.

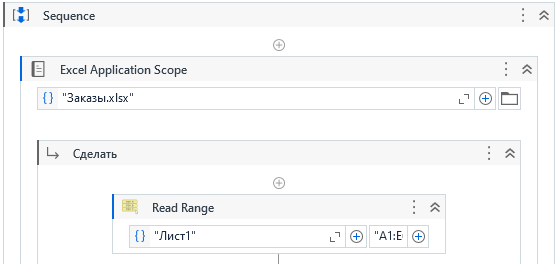


Рисунок 4.11 – Excel Application Scope

6. Добавить действие **For Each Row in DataTable**.

Указать **row** в секции **For Each** и переменную **DTable** в поле **DataTable**.

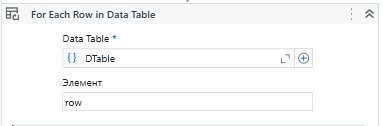


Рисунок 4.12 – For Each In DataTable

7. На ленте **Проект** нажать кнопку **Новая**, выбрать пункт **Последовательность** ив окне указать имя нового модуля, например, **ModulAnaliz**.

На панели **Аргументы** создать входные аргументы **in\_Payment** и **in\_Company** типа **String**.

8. Вернуться в проект **Main** и добавить в разделе **Body** действие **Invoke Workflow File**.

Внутри действия нажать кнопку с изображением папки и выбрать ранее созданную последовательность **ModulAnaliz**.

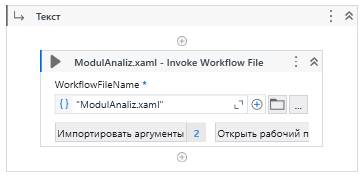


Рисунок 4.13 – Invoke Workflow File

Нажать **Импортировать аргументы**. Откроется окно **Аргументы вызываемого рабочего процесса (Invoked Workflow’s Arguments)**, где отображаются аргументы первой последовательности.

В поле **Имя** ввести **in\_Payment** со значением **row("Оплата").ToString** и во втором поле **Имя** − **in\_Company** со значением **row("Организация").ToString**.

Нажать **OK**. Аргументы импортируются.

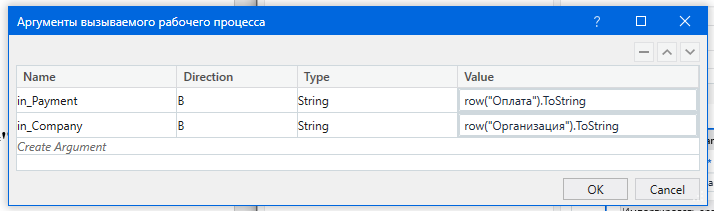


Рисунок 4.14 – Аргументы

9. Перейти к модулю **ModulAnaliz**.

Добавить действие [**If**](https://activities.uipath.com/docs/message-box).

В поле **Condition** записать выражение **in\_Payment ="Нет"**.

10. В блок **Then** добавить действие **Write Line**.

В поле **Text** ввести аргумент **in\_Company**.

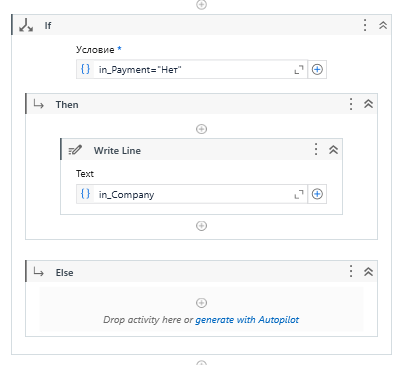


Рисунок 4.15 - ModulAnaliz

11. Вернуться в проект **Main** и запустить процесс автоматизации на выполнение. Проверить содержимое окна **Вывод**.

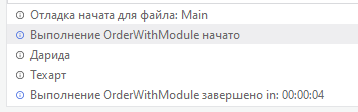


Рисунок 4.16 – результат

**Задание 4-4**. Создать проект, в котором данные получаются из одной таблицы Excel и добавляются в другой файл Excel. Затем опубликовать проект в библиотеке в особом канале, установить в другом проекте и использовать содержащиеся в нём действия.

1. Открыть UiPath Studio и выбрать пункт **Библиотека**, чтобы создать новый проект в библиотеке.

В появившемся окне **Новая пустая библиотека в**вести имя проекта, например, **LibraryFirst**. Добавить краткое описание, выбрать место расположения и нажать **Создать**.

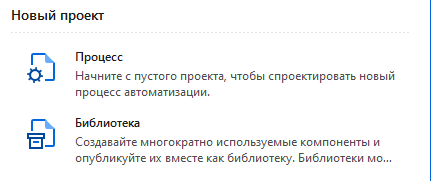


Рисунок 4.17 – Пункт для создания библиотеки

Новая библиотека откроется в Studio на панели **Проект**. Нажать **NewActivity.xaml**.

2. На панели **Аргументы** создать входной аргумент **in\_CurDir** типа **String** со значением по умолчанию **Directory.GetCurrentDirectory** (это необходимо для того, чтобы пути интерпретировались не относительно библиотеки, а относительно запускаемого проекта).



Рисунок 4.18 – Аргументы

3. Добавить действие **Read Range из раздела Система**.

В поле **Путь к рабочему журналу** добавить путь к файлу Excel, из которого нужно получить данные (например, **Path.Combine(in\_CurDir,“Examen.xlsx")**), при этом файл должен находиться в папке **LibraryFirst**, в противном случае надо прописать полный путь к файлу).

Указать имя листа, например, “**Лист1**”, а поле **Range** оставить пустым (если поле пустое, то читаются все ячейки).

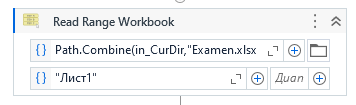


Рисунок 4.18 – Read Range Workbook

На панели свойств в поле **DataTable** при помощи сочетания клавиш **Ctrl+K создать** переменную **dtTable**. Автоматически будет создана переменная типа **DataTable.**

**В свойстве** **AddHeaders проверить наличие галочки. Если убрать галочку из свойства AddHeaders, то будет считываться текст вместе с заголовками.**

4. Добавить действие **Append Range** **из раздела Система**.

В поле **Путь к рабочему журналу** добавить в кавычках путь к файлу Excel, в который надо добавить данные, например, **Path.Combine(in\_CurDir,“Examen2.xlsx")**.

Указать имя листа, например, “**Лист1**”.

В поле **DataTable** добавить ранее созданную переменную **dtTable**.

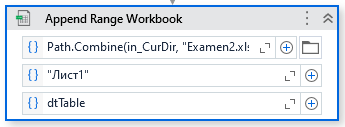


Рисунок 4.19 – Append Range Workbook

5. **Запустить** проект на выполнение.

Данные, собранные из первой таблицы Excel, добавляются в другую.

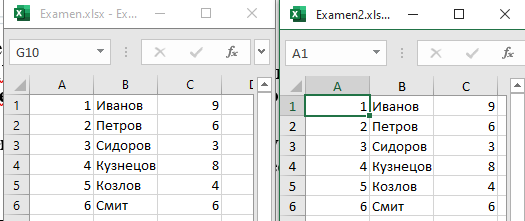


Рисунок 4.20 – Результат работы

**Публикация библиотеки в особый канал**

Чтобы использовать этот проект в других проектах автоматизации в качестве многократно используемого компонента, нужно преобразовать его в пакет (файл .**nupkg)**.

1. Открыть выше созданный проект в Studio.

2. Нажать кнопку **Опубликовать** на ленте **Проект**.

В появившемся окне на вкладке **Свойства пакета** можно привести краткое описание изменений, внесенных в проект, добавить новую версию в поле **Новая версия** и установить флажок **Предварительная версия**, чтобы пометить версию как alpha, если это необходимо.

На вкладке **Параметры публикации** в поле **Особый URL-адрес** ввести адрес папки, где будут храниться пакеты .**nupkg**.

Нажать кнопку **Опубликовать**. Если проект успешно опубликован, отобразится диалоговое окно **Информация**.

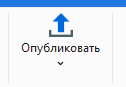


Рисунок 4.21 – Публикация библиотеки

**Задание 4-5**. Поместить другой проект из какой-либо лабораторной работы в библиотеку и опубликовать. Продемонстрировать использование получившегося проекта.

1. В настройках добавим пользовательский канал с библиотекой.

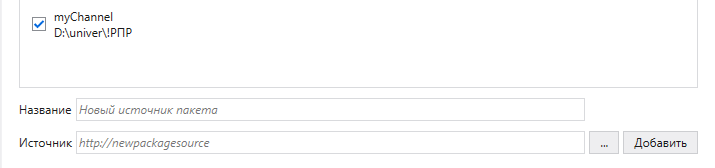


Рисунок 4.22 – Пользовательский канал

1. Установим библиотеку через Управление пакетами.

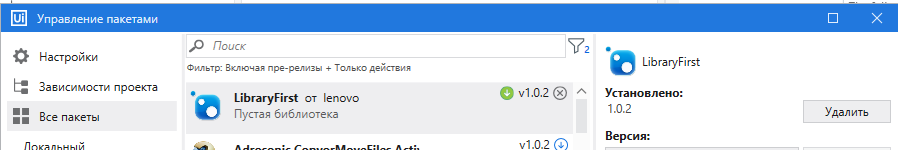


Рисунок 4.23 – Установка библиотеки

3. Добавить действие **Read Range из раздела Система**.

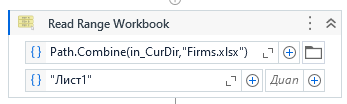


Рисунок 4.22 – Действие Read Range

4. Добавить действие **Append Range** **из раздела Система**.

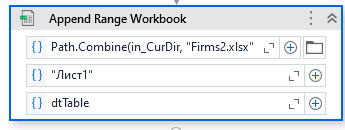


Рисунок 4.23 – Действие Append Range

Работа программы представлена на рисунке 4.24.

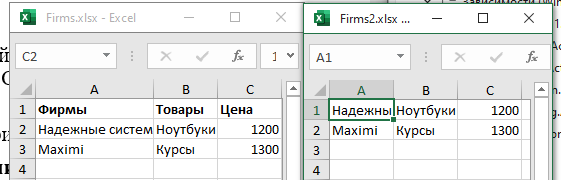


Рисунок 4.24 - Результат работы программы