МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 2-40 01 01 «Программное обеспечение

информационных технологий»

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: «Основы алгоритмизации и программирования»

**на тему: Разработка информационно-поисковой системы**

**«Курсы английского языка»**

Пояснительная записка

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Разработал |  |  |  |  | Закружный Е. С. |
|  | (дата) |  | (подпись) |  | (ФИО) |
| Руководитель |  |  |  |  | Глушенок А. В. |
|  | (дата) |  | (подпись) |  | (ФИО) |

Гомель, 2022

СОДЕРЖАНИЕ

[Введение…………………………………………………………………3](#_Toc106823536)

[1 Подпрограммы. Процедуры и функции……………………4](#_Toc106823537)

[2 ОПИСАНИЕ АЛГОРИТМОВ И ТЕХНОЛОГИИ РЕШЕНие………...8](#_Toc106823538)

[2.1 Постановка задачи и исходные данные 8](#_Toc106823539)

[2.2 Описание работы процедур 9](#_Toc106823540)

[2.3 Организация диалога с пользователем 10](#_Toc106823541)

[3 ИСПОЛНЕНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ…………………………..14](#_Toc106823542)

[3.1 Описание интерфейса 14](#_Toc106823543)

[3.2 Программная реализация подпрограмм 18](#_Toc106823544)

[4 результат работы…………………………………………………..24](#_Toc106823545)

[Заключение.............................................................................................38](#_Toc106823546)

[Список использованных источников…………...……….............39](#_Toc106823547)

[ПРИЛОЖЕНИЕ А………………………………………………………….40](#_Toc106823548)

# ВведениЕ

Целью проекта является разработка информационно-поисковой системы «Курсы английского языка». Программа позволит упростить работу пользователя с информацией о преподавателях, учениках и курсах.

Информационно-поисковая система — это система, которая позволяет искать, обрабатывать, отбирать требуемые данные запроса в своей особой базе, где находятся описания различных источников информации, а также правила пользования ими. Основной ее задачей является поиск нужной пользователю информации. Для того, чтобы он был более эффективным, используется понятие релевантности, то есть то, насколько сами результаты поиска точно подходят тому или иному запросу.

Поставленные задачи:

* выполнить теоретическое задание;
* организовать информационную базу проекта;
* описать алгоритмы и технологии решения;
* разработать интерфейс программы;
* обеспечить создание нового файла;
* разработать стандартный функционал;
* разработать функции индивидуального задания;
* организовать проверку на корректность ввода;
* протестировать приложение;
* написать пояснительную записку.

Результатами выполнения проекта является готовая информационно-поисковая система «Курсы английского языка».

Пользователь сможет легко ориентироваться в системе, а также обрабатывать данные файла. Система используется для оптимизации учебного процесса. Сама информационно-поисковая система будет разработана на языке «Delphi».

«Delphi» — это язык программирования общего назначения, который использует диалект Delphi объектного языка программирования «Pascal» и предоставляет интегрированную среду разработки (IDE) для быстрой разработки приложений настольного, мобильного, веб- и консольного программного обеспечения.

«Delphi» позволяет создавать приложения для Windows в пять раз быстрее, чем другие решения для разработки, или тратить на это в пять раз меньше ресурсов, не жертвуя производительностью и возможностями.

# 1 Подпрограммы. Процедуры и функции

Подпрограмма — это именованная часть программы, представляющая собой некоторое собрание операторов, структурированных аналогично основной программе.

Описываются подпрограммы между разделом описания переменных и началом тела основной программы. Структура всей программы в «Pascal» приведена на рисунке 1.1.

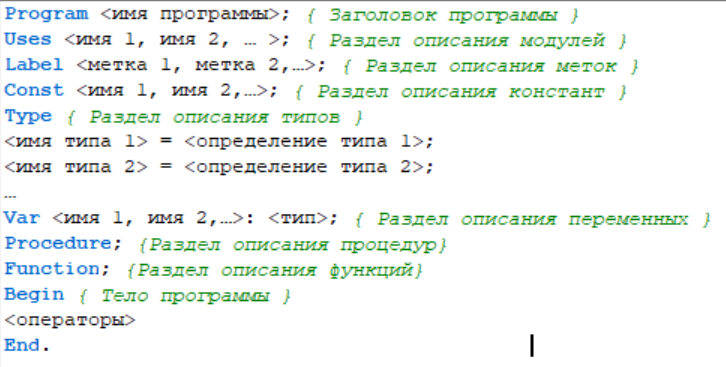


Рисунок 1.1 — Структура программы в «Pascal»

Подпрограммы в «Pascal» представлены в виде функций и процедур, которые могут бытьвстроенными и пользовательскими*.*

Встроенные подпрограммы — это подпрограммы, которые изначально присутствуют в системе.

Пользовательские подпрограммы — это подпрограммы, которые создаются программистом.

В «Pascal» переменные по отношению к программе делятся на **локальные** и **глобальные**.

Глобальными переменными называют переменные, которые объявляются в основной части.

Локальными переменныминазывают переменные, которые объявляются в процедуре или функции. Локальные переменные в отличие от глобальных могут использоваться лишь внутри подпрограммы, к которой они принадлежат.

Для передачи данных, после объявления имени подпрограммы, в круглых скобках необходимо указать параметры, которые в свою очередь можно разделить на передаваемые и возвращаемые.

Передаваемые параметры — это переменные, которые подпрограмма получает из главной части программы, а возвращаемые — отдает (возвращает).

Различают **формальные** и **фактические** параметры.

Фактическими параметрами называют параметры, которые используются при вызове процедуры или функции.

Формальные параметраминазывают параметры,которыеописываются в заголовке подпрограммы, и принимают значения фактических параметров. Формальные параметры должны совпадать типом, количеством и порядком следования с фактическими.

Увидеть, где находятся и описываются формальные и фактические параметры в структуре процедур и функций, можно на рисунке 1.2 и 1.3.

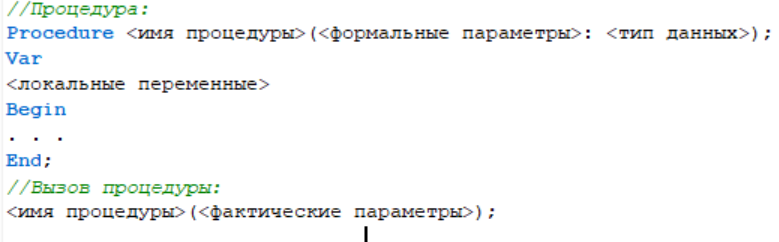


Рисунок 1.2 — Пример структуры процедуры

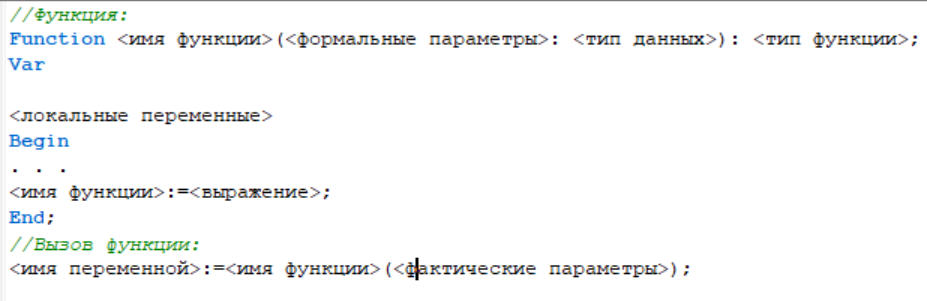


Рисунок 1.3 — Пример структуры функции

Процедура — это независимая совокупность операторов, выполняющих заданную последовательность действий, и определенных отдельно от основной части программы. Она начинается со служебного слова «Procedure», после которого идет имя и заголовок процедуры. Остальная часть аналогична основной программе.

Пример процедуры представлен на рисунке 1.4.

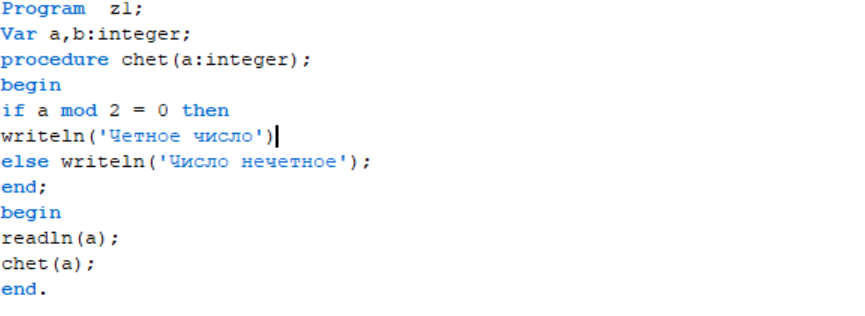


Рисунок 1.4 — Пример процедуры

Для обращения к процедуре используется оператор вызова процедуры. Он состоит из идентификатора (имени) процедуры и списка фактических параметров, отделённых от друг друга запятыми и заключённых в круглые скобки. Список параметров может отсутствовать, если процедуре не передаётся никаких значений.

Формат обращения к процедуре: «имя\_процедуры(фактические\_параметры)».

Процедура может возвращать сразу несколько результатов. Пример приведён на рисунке 1.5.



Рисунок 1.5 — Пример процедуры с различными фактическими параметрами

В отличие от процедур, пользовательские функции могут быть задействованы в основной программе, лишь в каком-либо выражении. Для обращения к функции необходимо знать тип данных возвращаемого ей результата.

Исходя из рисунка 1.3, описание функций начинается со служебного слова «function», вслед за которым идет ее заголовок, а дальше те же блоки, что и у процедуры.

В вызываемую часть результат работы функции должен быть возвращен посредством имени последней. Это предусматривает хотя бы одно присвоение ей значения.

Пример функции представлен на рисунке 1.6.

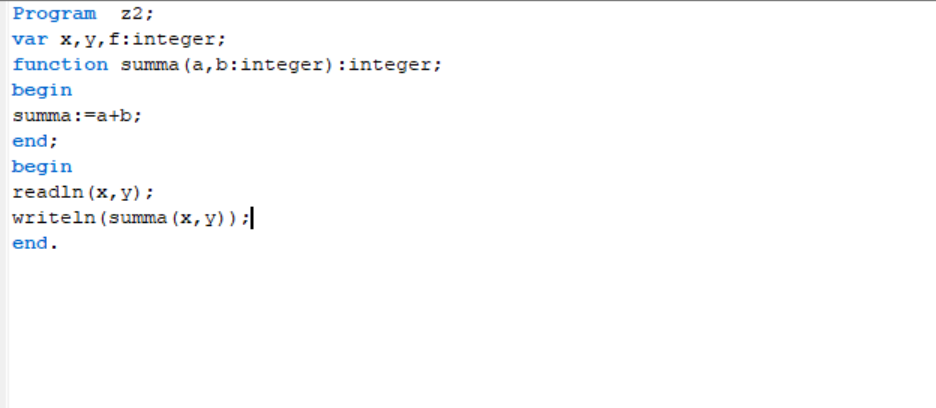


Рисунок 1.6 — Пример функции

В разделе операторов должен находиться по крайней мере один оператор, присваивающий идентификатор функции значение. Если таких присваиваний несколько, то результатом выполнения функции будет значение последнего оператора присваивания.

Существует 2 способа вызова функции. Через команду «writeln» и через присваивание переменной значение функции. Пример функции, вызванной двумя способами представлен на рисунке 1.7.

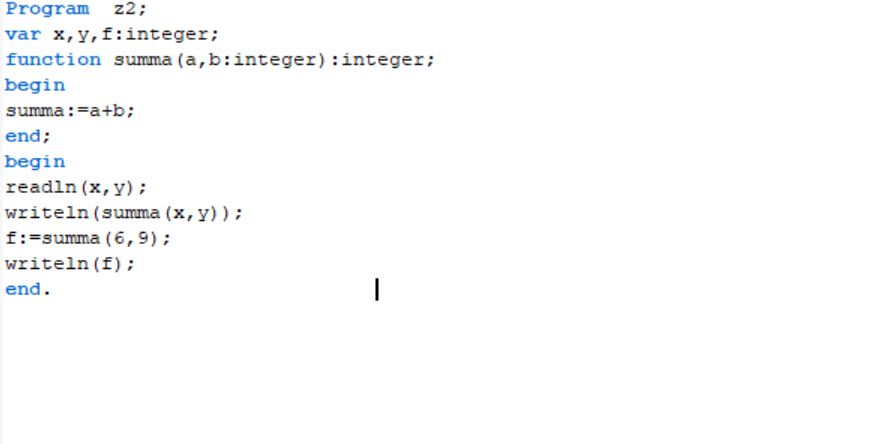


Рисунок 1.7 — Функция, вызванная двумя способами

# 2 ОПИСАНИЕ АЛГОРИТМОВ И ТЕХНОЛОГИИ РЕШЕНИЯ

## 2.1 Постановка задачи и исходные данные

Программа должна содержать следующие функции: открытие и вывод содержимого файла, сохранение созданного пользователем файла на компьютер, добавление группы в файл и последующее ее сохранение в файле, удаление выбранной пользователем группы, удаление группы, где учащихся меньше 10 человек , увеличение стоимости обучения для определенного уровня владения языка, вывод групп занимающихся в определенный день недели, сортировка групп по всем критериям по возрастанию и убыванию, редактирование групп непосредственно в самой таблице, а также поиск по символам.

В программе для записи данных необходимо использовать типизированный файл.

Типизированный файл — это файл, с объявленным типом его компонентов, в котором содержатся данные одного типа и фиксированной длины.

Для того, чтобы открыть файл как типизированный, его необходимо ассоциировать с файловой переменной типа запись.

Запись — это структурированный тип данных, состоящий из фиксированного числа компонент (полей) разного типа.

Запись использования хранения данных курсового проекта «Курсы английского языка» называется «groupRecord». Все атрибуты и их запись представлены на рисунке 2.1.

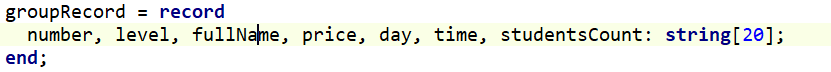


Рисунок 2.1 — Описание записи

Описание атрибутов записи:

* первый атрибут под названием «number» содержит информацию о номере группы;
* второй атрибут под названием «level» содержит информацию о уровне владения языком группы;
* третий атрибут под названием «fullName» содержит информацию о ФИО преподавателя;
* четвёртый атрибут под названием «price» содержит информацию о стоимости занятия;
* пятый атрибут под названием «day» содержит информацию о дне проведения курсов;
* шестой атрибут под названием «time» содержит информацию о времени проведения курсов;
* седьмой атрибут под названием «studentsCount» содержит информацию о количестве учащихся на курсах.

Все атрибуты описаны в строковом типе «string» в диапазоне двадцать.

## 2.2 Описание работы процедур

В разработке информационно-поисковой системы «Курсы английского языка» на языке программирования «Delphi» выполнен стандартный функционал и функционал по заданию.

Для открытия файла нужно перейти на основную форму, нажать в верхнем меню на пункт «Файл», после чего в открывшимся меню нажать на пункт «Открыть». Появится проводник и открыть в нем нужный файл, ранее сохраненный на этом компьютере.

Сохранение происходит аналогично через открытие пункта «Файл» на основной форме и выбора пункта в открывшемся меню под названием «Сохранить как». Также открывается проводник, пользователь может указать название файла и сохранить в выбранное место на своем компьютере.

Сохранение данных таблицы происходит в том же меню, только при выборе пункта «Сохранить». Изменения сохраняются в файл.

Добавление группы происходит по нажатию на форме на кнопку «Добавить группу». После этого появляется новая форма, где указываются все данные о группе, после заполнения пользователь нажимает на кнопку на новой форме «Добавить группу». Она появляется в таблице групп и пользователю предлагается перед работой с таблицей групп, сохранить ее файлом на своем компьютере во избежание потери данных таблицы.

Удаление группы происходит через нажатие пользователем ПКМ по строке таблицы. После появляется уведомление с вопросом к пользователю о удалении, при положительном ответе удаление группы происходит без сохранения в файле.

Отмена изменений происходит через кнопку на верхнем меню «Отменить изменения».

Удаление групп, где меньше 10 человек реализовано кнопкой в пункте «Функции» в верхнем меню.

Редактировать данные в таблице можно непосредственно в самой таблице, кликнув на текст атрибута двойным кликом, появится новая форма для внесения изменений. Сделав редактирование атрибута, пользователь должен нажать на «Подтвердить», после чего изменения сохранятся в таблице.

Сортировка по возрастанию и убыванию по выбранному атрибуту происходит через кнопку на верхнем меню «Сортировка». Кликнув на кнопку «Сортировка» откроется меню, где пользователь может выбрать по какому атрибуту отсортировать таблицу, а также наведясь на атрибут пользователь сможет выбрать по возрастанию или по убыванию отсортировать таблицу.

Фильтрацию по дню проведения курсов происходит на основной форме по нажатию на кнопку в верхнем меню «Фильтрация» и выбрав пункт «Дата проведения». На открывшейся форме пользователь может выбрать день недели и по нажатию на кнопку «Применить» таблица фильтруется.

Увеличение стоимости занятий определённому уровню языка происходит на основной форме, где нужно кликнуть в верхнем меню на кнопку «Функции» и выбрать пункт «Увеличить стоимость занятий». После этого на открывшейся форме выбрать уровень языка, для которого нужно увеличить стоимость занятий и нажать на кнопку «Применить».

Поиск по символам реализован на основной форме, в верхней части формы находится строка, в которую пользователь может вводить символы и, если эти символы есть в таблице они будут выделяться.

## 2.3 Организация диалога с пользователем

Перечень используемых компонентов для взаимодействия с пользователем:

* «Button»;
* «Edit»;
* «MainMenu»;
* «StringGrid»;
* «DateTimePicker»;
* «OpenDialog»;
* «ListBox».

Компонент «Button» используется во всех формах программы. Он используется для открытия другой формы, открытия функций программы, для применения изменений на вспомогательных формах.

Компонент «Edit» используется на некоторых формах программы. Он используется для ввода различной информации. Такой как: ввод символов в поиске, ввод номера группы, уровня языка, ФИО преподователя, стоимости за занятие и количества учащихся в вспомогательной форме «Добавить группу».

Компонент «MainMenu» используется для вызова функций программы и для работы с файлом, а также для удаления выбранной строки.

Компонент «StringGrid» используется, как главный компонент на основной форме для хранения и редактирования в нем данных.

Компонент «DateTimePicker» используется для введения даты и времени на вспомогательной форме «Добавить группу».

Компонет «OpenDialog» используется для сохранения и открытия файлов на компьютере.

Компонент «ListBox» используется на двух вспомогательных формах для выбора определенного дня недели и уровня владения языком.

Спроектированный макет начальной формы представлен на рисунке 2.2.

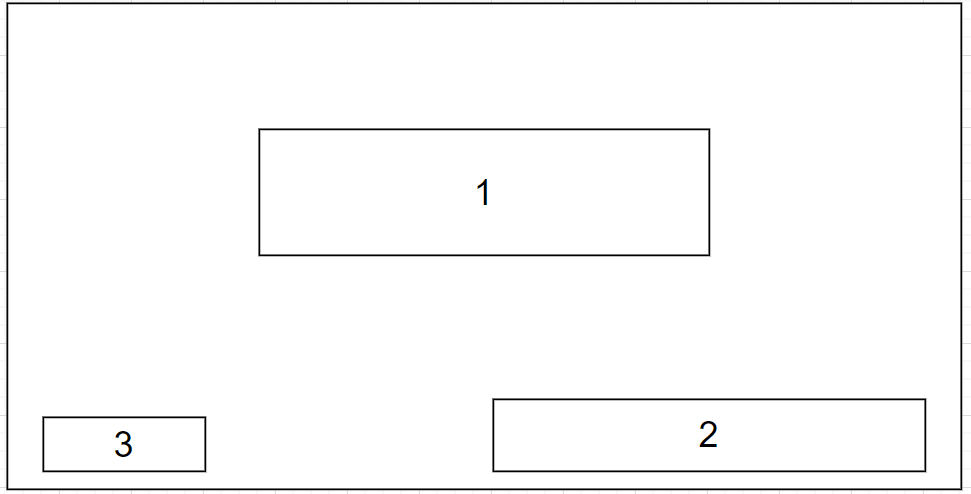


Рисунок 2.2 — Макет начальной формы: 1 — блок с названием работы; 2 — блок с автором продукта; 3 — блок-переход

Графический интерфейс основной формы представлен на рисунке 2.3.

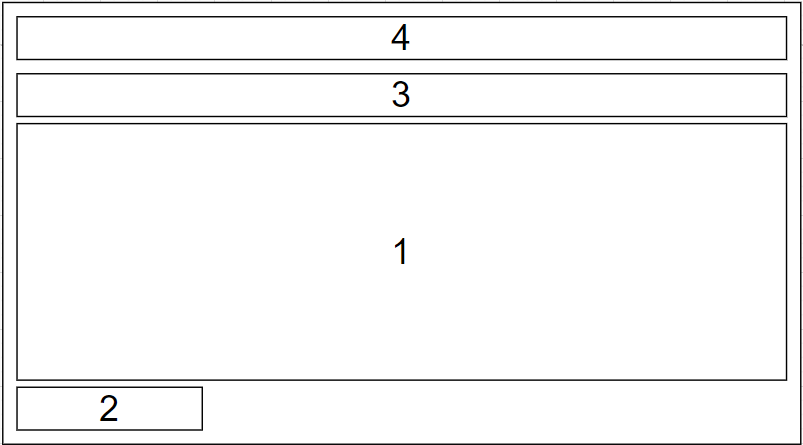


Рисунок 2.3 — Макет основной формы: 1 — блок-таблица для работы с данными; 2 — блок добавления; 3 — блок поиска по символам; 4 — блок главного меню

Макет формы добавления группы представлен на рисунке 2.4.

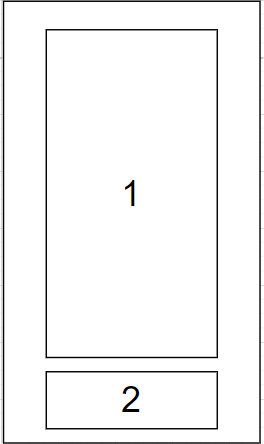


Рисунок 2.4 — Макет формы добавления группы: 1 — блок добавления номера группы, уровня владения языком, ФИО преподавателя, стоимости за занятие, дня проведения, времени проведения, количества учащихся; 2 — блок подтверждения окончательного добавления группы

Макет формы фильтрации по дню проведения представлен на рисунке 2.5.

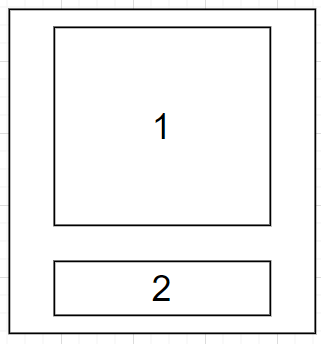


Рисунок 2.5 — Макет формы фильтрации по дню проведения: 1 — блок выбора дня недели; 2 — блок применения изменений

Макет формы увеличения стоимости занятий представлен на рисунке 2.6.

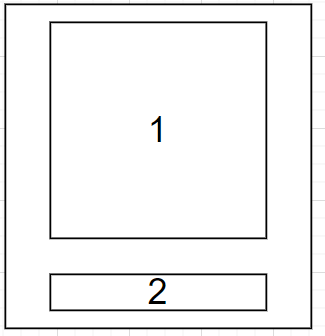


Рисунок 2.6 — Макет формы увеличения стоимости занятий: 1 — блок выбора уровня владения языком; 2 — блок применения изменений

Макет формы фильтрации по дате проведения представлен на рисунке 2.7.

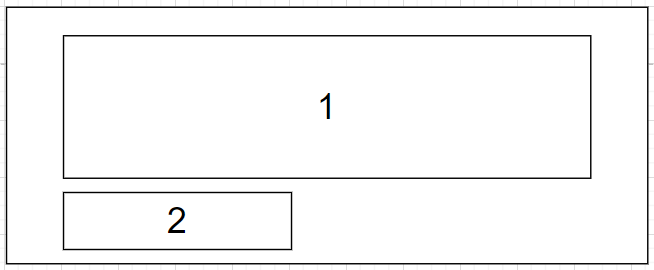


Рисунок 2.7 — Макет формы фильтрации по дате проведения: 1 — блок задачи даты проведения; 2 — блок применения фильтрации

# 3 ИСПОЛНЕНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ

## 3.1 Описание интерфейса

Интерфейс — это «проводник» между человеком и программой, операционной системой, техническим устройством или способ взаимодействия приложений между собой.

Графический пользовательский интерфейс — это система средств для взаимодействия пользователя с электронными устройствами, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана.

Данная программа состоит из 6 форм. Первая форма «Начальная» обозначает начало самой программы, на ней находятся данные о авторе и названии продукта, а также кнопка для перехода на основную форму для работы с программой.

Первая форма программы представлена на рисунке 3.1.

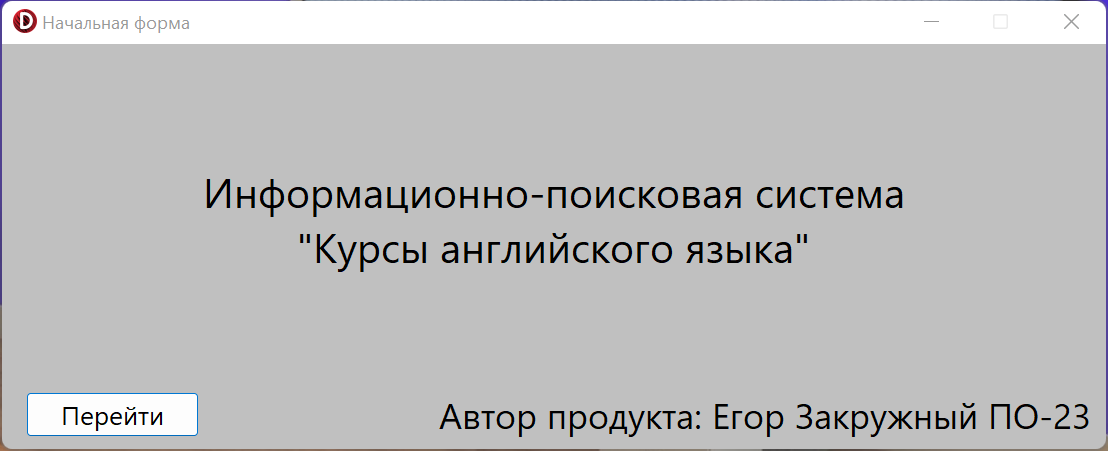


Рисунок 3.1— «Начальная» форма

Вторая форма программы, которая называется «Основная», содержит «Stringgrid» —таблица для работы пользователя с файлом, кнопку для добавления групп в таблицу «Добавить группу». На форме присутствует «Edit» для поиска данных в таблице.

Также форма содержит компонент «MainMenu», в котором находятся пункты для работы с файлом: «Файл», «Сортировка», «Фильтрация», «Функции», «Отменить изменения». Командная кнопка «Добавить группу» открывает форму для добавления группы в таблицу. Удаление реализовано на ПКМ, при нажатии по таблице появляется форма с подтверждением удаления выбранной строки, если строка не выбрана удаляется первая строка таблицы. Редактирование реализовано двойным кликом по ячейке таблицы.

При нажатии на «Функции» в компоненте «MainMenu», который находится на основной форме, открываются пункты «Увеличить стоимость занятий на 10% для определенного уровня языка», «Вывести информацию о группах, занимающихся в определенный день недели», «Удалить группы, где учащихся меньше 10 человек». Пункт «Удалить группы, где учащихся меньше 10 человек» удаляет из таблицы группы, где учащихся меньше 10 человек.

При нажатии на «Файл» в компоненте «MainMenu», который находится на основной форме, открываются пункты «Открыть», «Сохранить», «Сохранить как».

Выбрав пункт «Открыть», пользователь сможет открыть сохранённый файл, с которым он уже работал. Пункт «Сохранить» сохраняет изменения в файле, если файл уже сохранен на компьютере, а если нет, то появится диалоговое окно для сохранения файла. Пункт «Сохранить как» вызывает диалоговое окно для сохранения файла.

При нажатии на «Сортировка» в компоненте «MainMenu», который находится на основной форме, открываются пункты «Номер группы», «Уровень», «ФИО преподавателя», «Стоимость за занятие», «Дата проведения», «Количество учащихся». По каждому из этих атрибутов можно отсортировать таблицу по возрастанию и убыванию.

«Основная» форма представлена на рисунке 3.2.

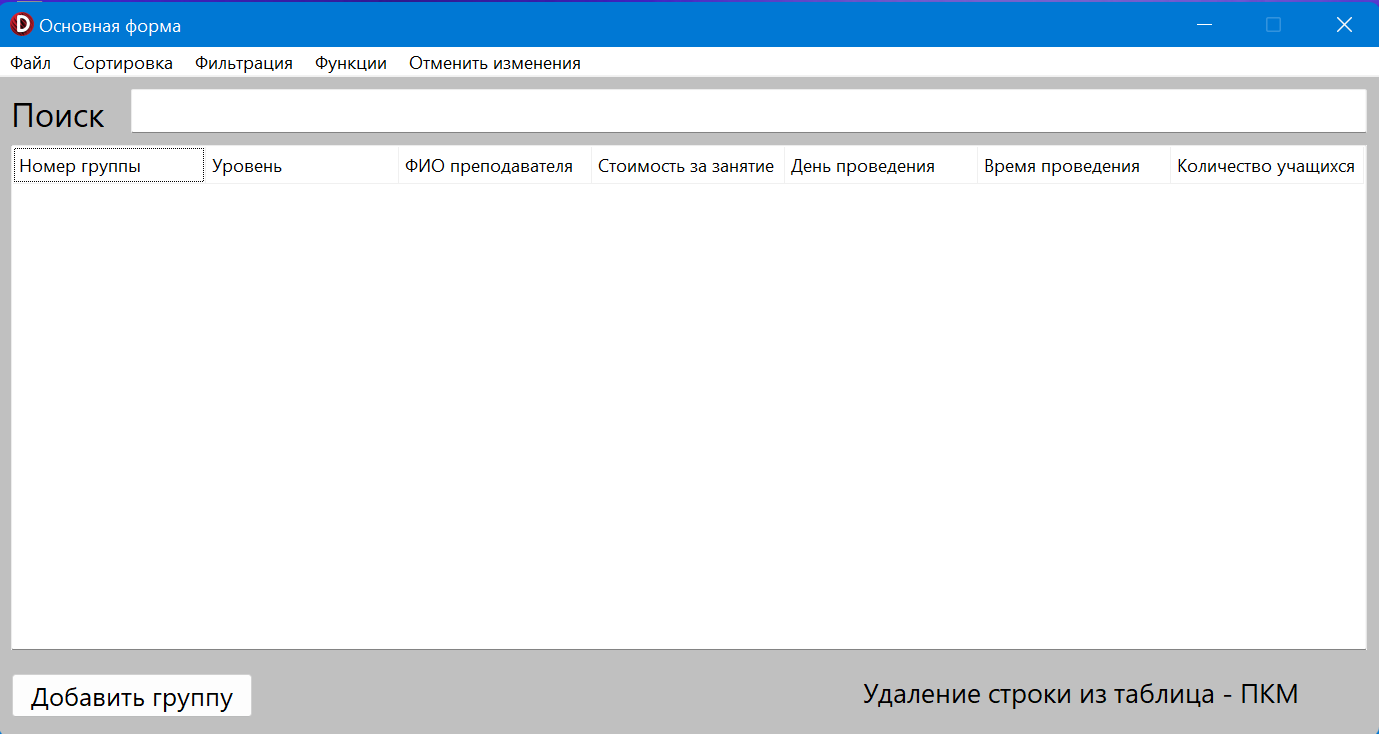


Рисунок 3.2 —«Основная» форма

При нажатии на кнопку «Добавить группу» появляется форма, где пользователь должен заполнить атрибуты группы и добавить группу в таблицу.

Форма «Добавить группу» представлена на рисунке 3.3.

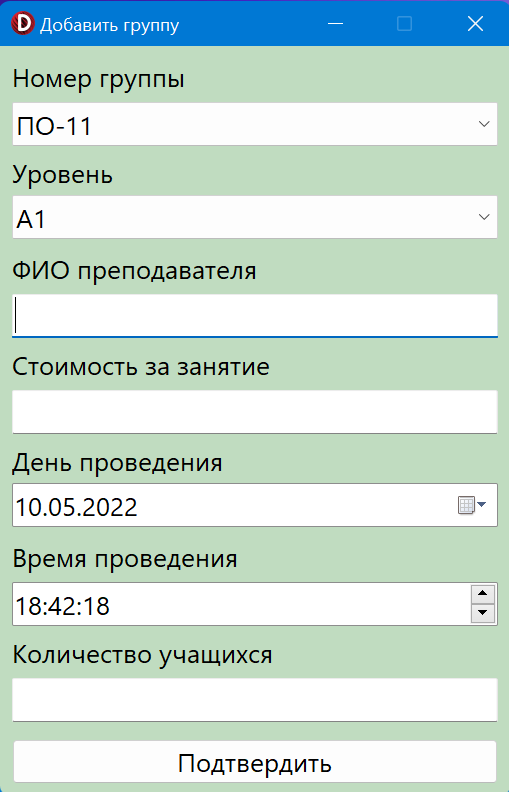


Рисунок 3.3 — Форма «Добавить группу»

В разделе «Функции» пункт «Вывести информацию о группах, занимающихся в определенный день недели» вызывает форму с выбором дня недели и применения последующих изменений.

Форма «Фильтрация по дню проведения» представлена на рисунке 3.4.

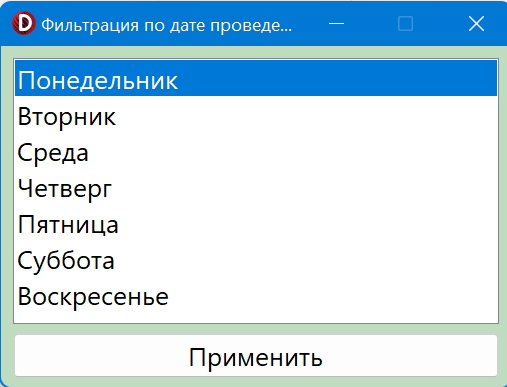


Рисунок 3.4 — Форма «Фильтрация по дню проведения»

В разделе «Функции» пункт «Увеличить стоимость занятий на 10% для определенного уровня языка» вызывает форму с выбором уровня языка и применения последующих изменений.

Форма «Увеличение стоимости занятий» представлена на рисунке 3.5.

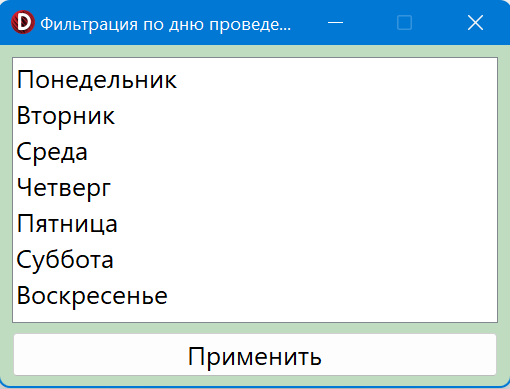


Рисунок 3.5 — Форма «Увеличение стоимости занятий»

При нажатии на «Фильтрация» в компоненте «MainMenu», который находится на основной форме, открываются пункты «Номер группы», «Уровень», «ФИО преподавателя», «Стоимость за занятие», «Дата проведения», «Количество учащихся». По каждому из этих атрибутов можно отфильтровать таблицу.

Форма фильтрации по атрибуту «Дата проведения» представлен на рисунке 3.6.

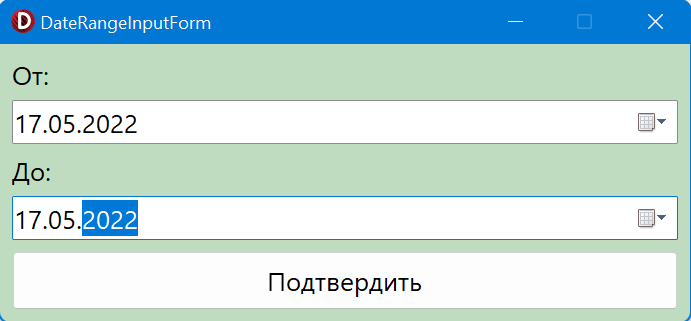


Рисунок 3.6 — Форма фильтрации по атрибуту «Дата проведения»

Удаление реализовано на ПКМ, при нажатии по таблице появляется уведомление с подтверждением удаления выбранной строки, если строка не выбрана удаляется первая строка таблицы. Уведомление «Подтверждение удаления» представлена на рисунке 3.7.

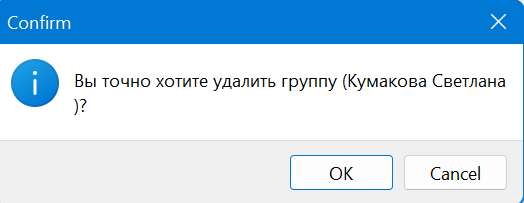


Рисунок 3.7 — Уведомление «Подтверждение удаления»

## 3.2 Программная реализация подпрограмм

Код программы представляет собой совокупность процедур, связанных между собой. В программе реализован стандартный функционал и функционал по заданию.

На рисунке 3.8 представлена фрагмент кода для открытия файла.

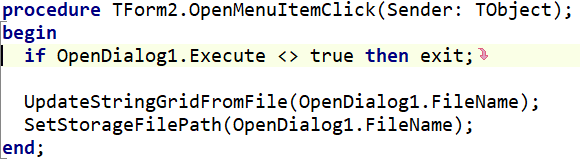


Рисунок 3.8 — Фрагмент кода для открытия файла

На рисунке 3.9 представлена фрагмент кода для сохранения открытого файла.

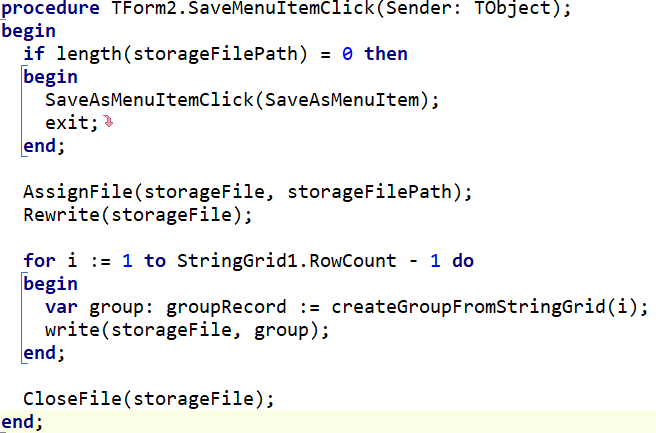


Рисунок 3.9 — Фрагмент кода для сохранения открытого файла

На рисунке 3.10 представлена фрагмент кода для сохранения нового файла.

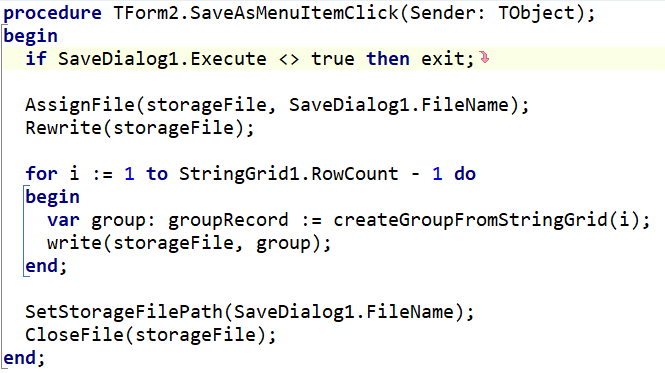


Рисунок 3.10 — Фрагмент кода для сохранения нового файла

На рисунке 3.11 представлена фрагмент кода для символьного поиска.

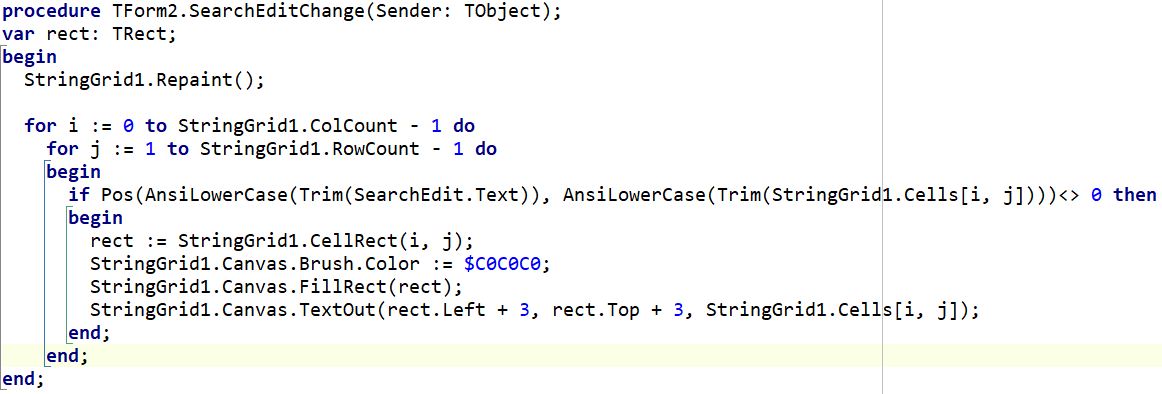


Рисунок 3.11 — Фрагмент кода для символьного поиска

На рисунке 3.12 представлена часть кода для удаления группы из таблицы.

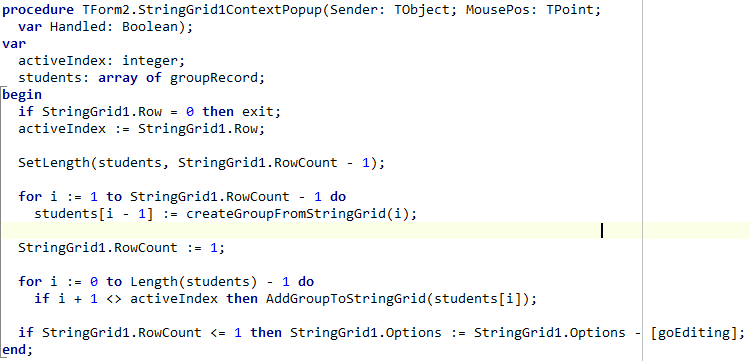


Рисунок 3.12 — Фрагмент кода для удаления группы из таблицы

На рисунке 3.13 представлена фрагмент кода редактирования атрибутов группы из таблицы.

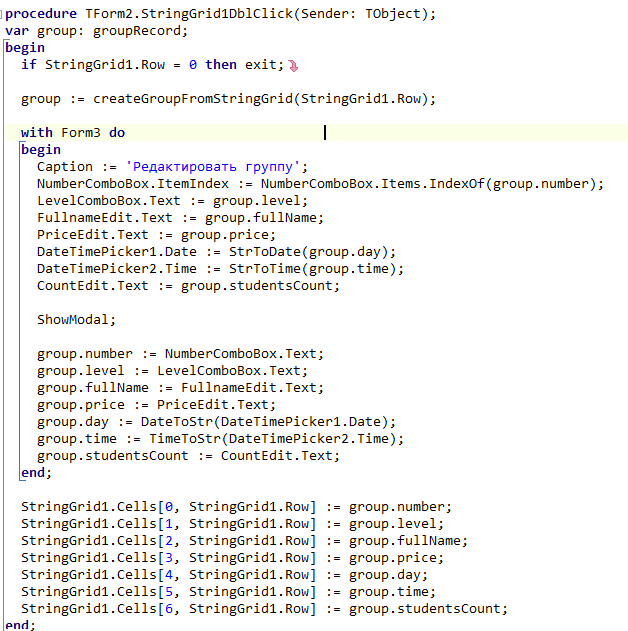


Рисунок 3.13 — Фрагмент кода для редактирования атрибутов группы из таблицы

На рисунке 3.14 представлена фрагмент кода для добавления группы в таблицу.

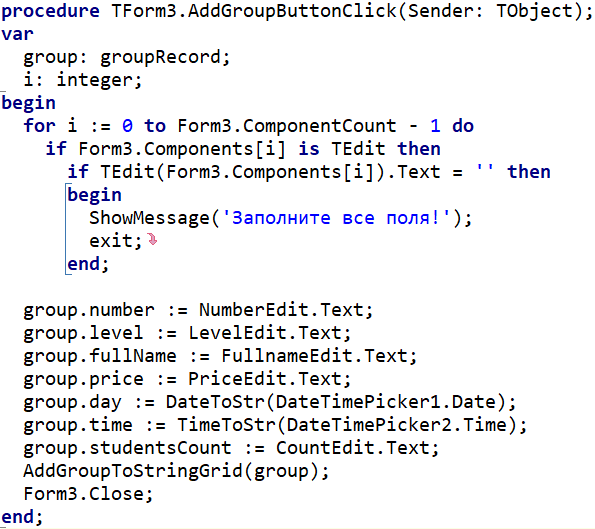


Рисунок 3.14 — Фрагмент кода для добавления группы в таблицу

На рисунке 3.15 представлена фрагмент кода для сортировки таблицы по возрастанию по одному из атрибутов.

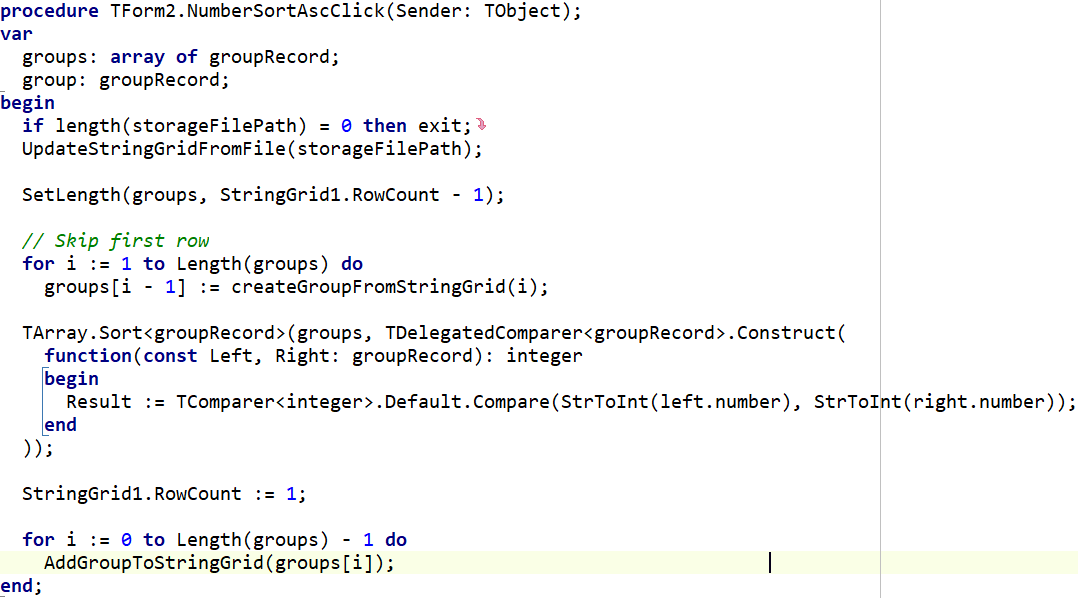


Рисунок 3.15 — Фрагмент кода для сортировки таблицы по возрастанию по одному из атрибутов

На рисунке 3.16 представлена фрагмент кода для сортировки таблицы по убыванию по одному из атрибутов.

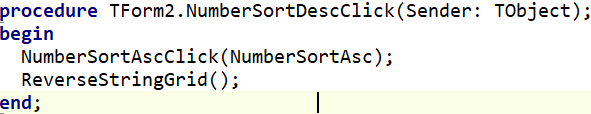


Рисунок 3.16 — Фрагмент кода для сортировки таблицы по убыванию по одному из атрибутов

На рисунке 3.17 представлена фрагмент кода для фильтрации таблицы по одному из атрибутов.

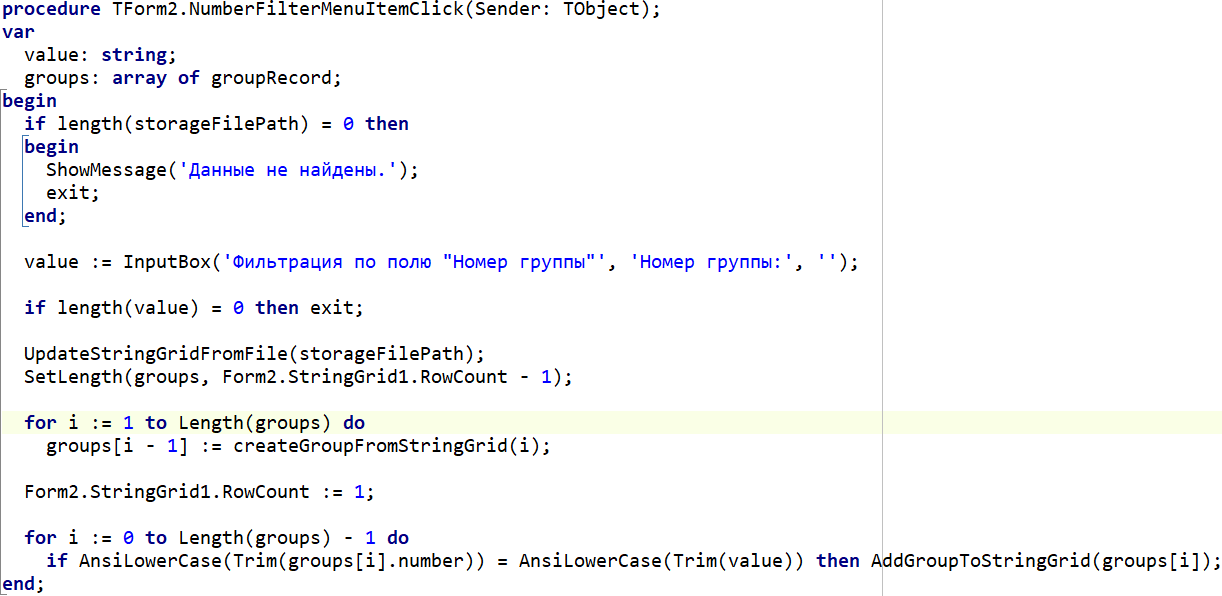


Рисунок 3.17 — Фрагмент кода для фильтрации таблицы по одному из атрибутов

На рисунке 3.18 представлена фрагмент кода для увеличения стоимости занятий на 10% для групп определенного уровня.

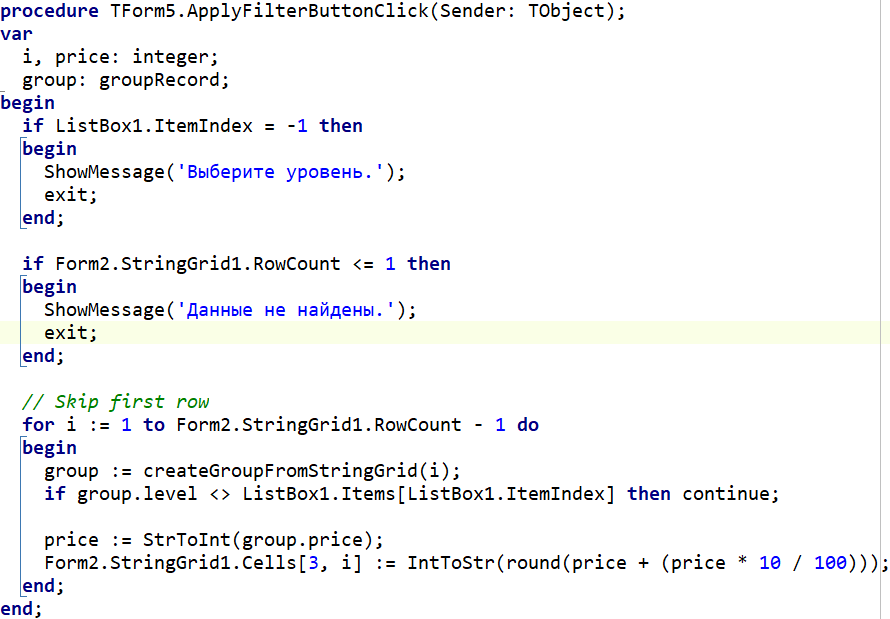


Рисунок 3.18 — Фрагмент кода для увеличения стоимости занятий на 10% для групп определенного уровня

На рисунке 3.19 представлена фрагмент кода для вывода информации о группах, занимающихся в определенный день недели.

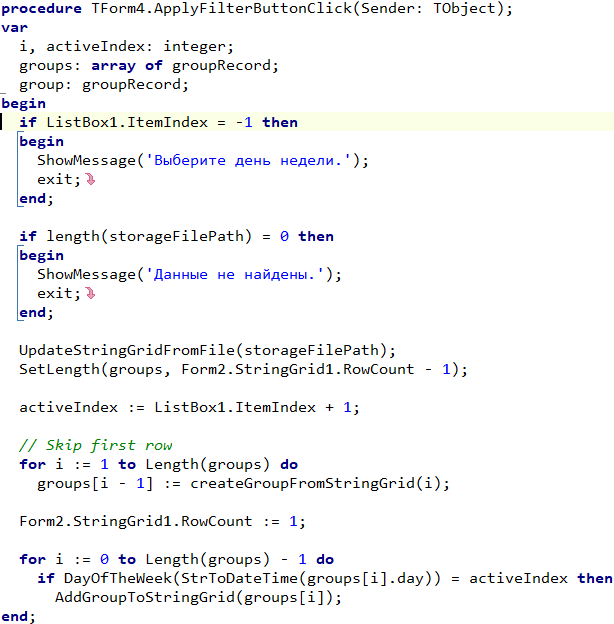


Рисунок 3.19 — Фрагмент кода для вывода информации о группах, занимающихся в определенный день недели

На рисунке 3.20 представлена фрагмент кода для удаления групп, где количество учащихся меньше 10.

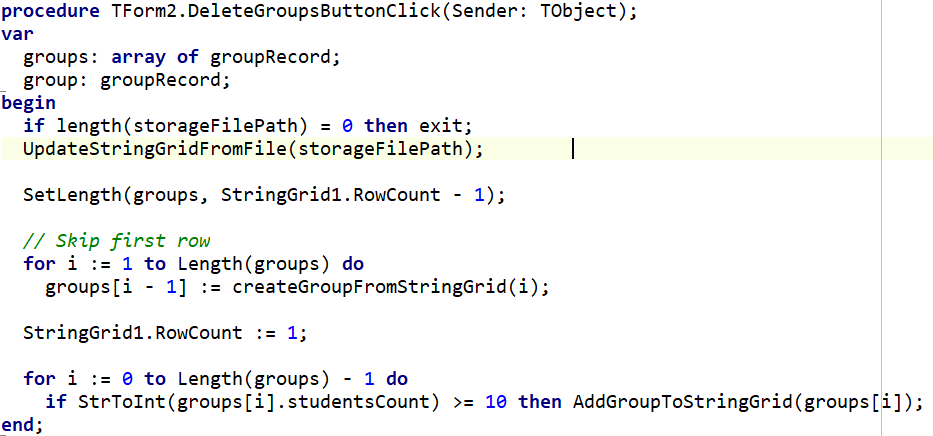


Рисунок 3.20 — Фрагмент кода для удаления групп, где количество учащихся меньше 10

Полный код программы представлен в приложении А.

# 4 результат работы

При запуске программы появляется форма под названием «Титульная». На данной форме присутствует следующая информация: тема курсового проекта, ФИО и группа разработчика, кнопка для перехода на основную форму.

«Титульная» форма отображена на рисунке 4.1.

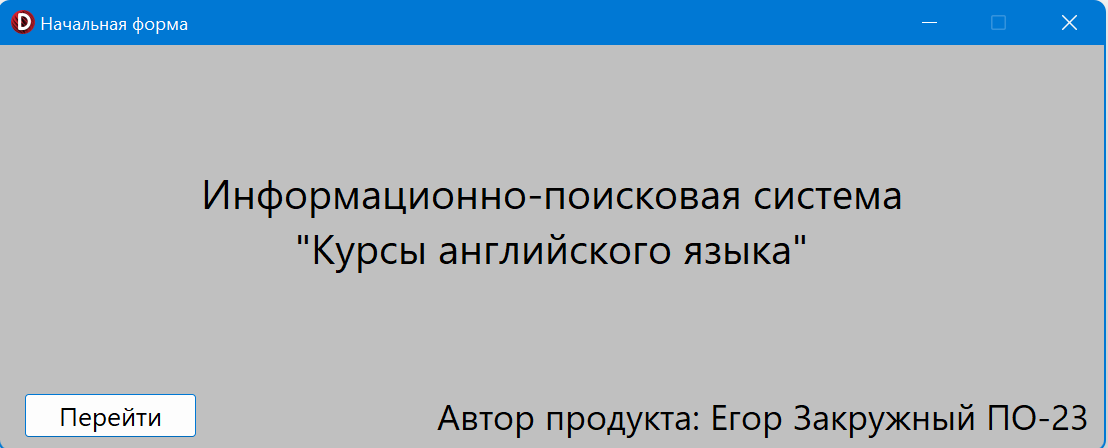


Рисунок 4.1 ⎯ «Титульная» форма

Нажав на кнопку «Перейти», пользователь переходит на «Основную» форму программы. Пользователю становятся доступны все основные функции программы.

«Основная» форма отображена на рисунке 4.2.

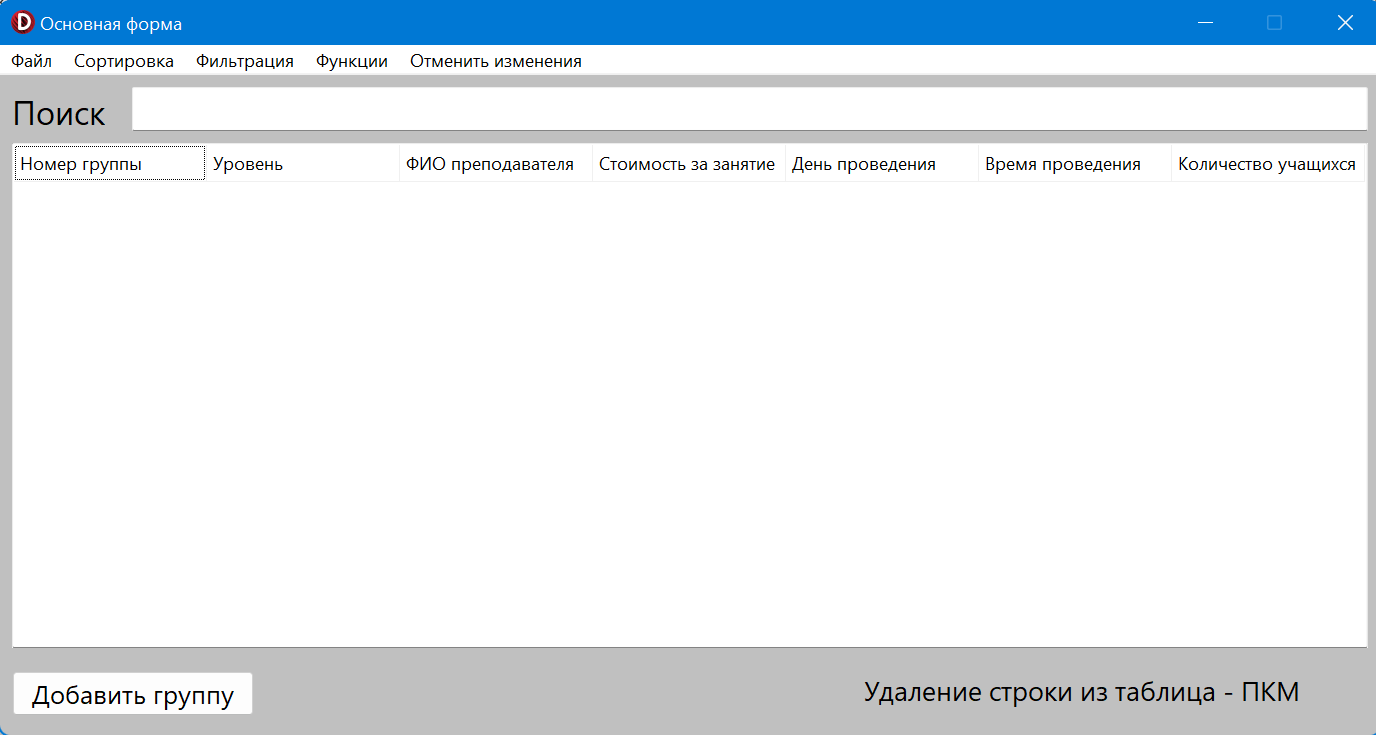


Рисунок 4.2 ⎯ «Основная» форма

На основной форме пользователь может открыть уже ранее сохраненный файл, с помощью кнопки в меню «Файл» и выбрать пункт «Открыть», после чего откроется диалоговое окно.

Открытие файла представлено на рисунке 4.3.

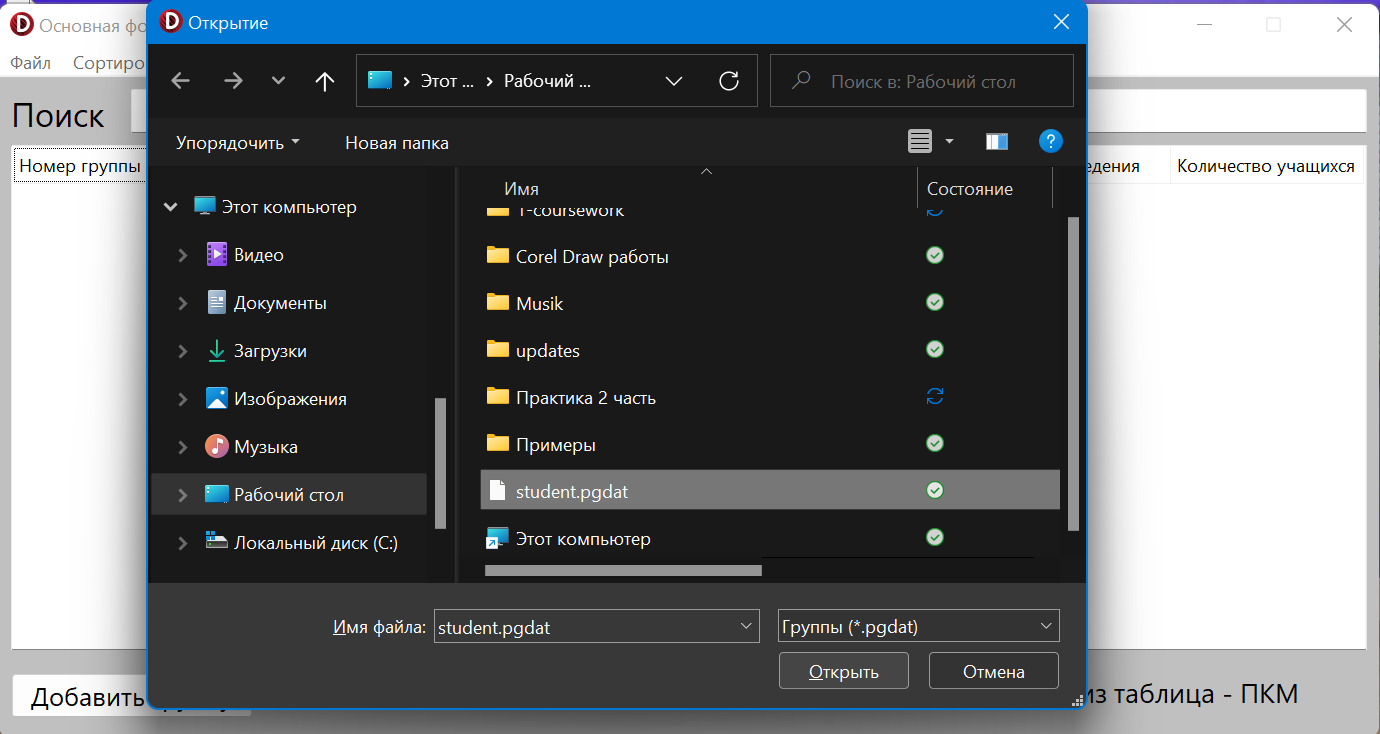


Рисунок 4.3 ⎯ Открытие файла

Результат открытия файла представлен на рисунке 4.4.

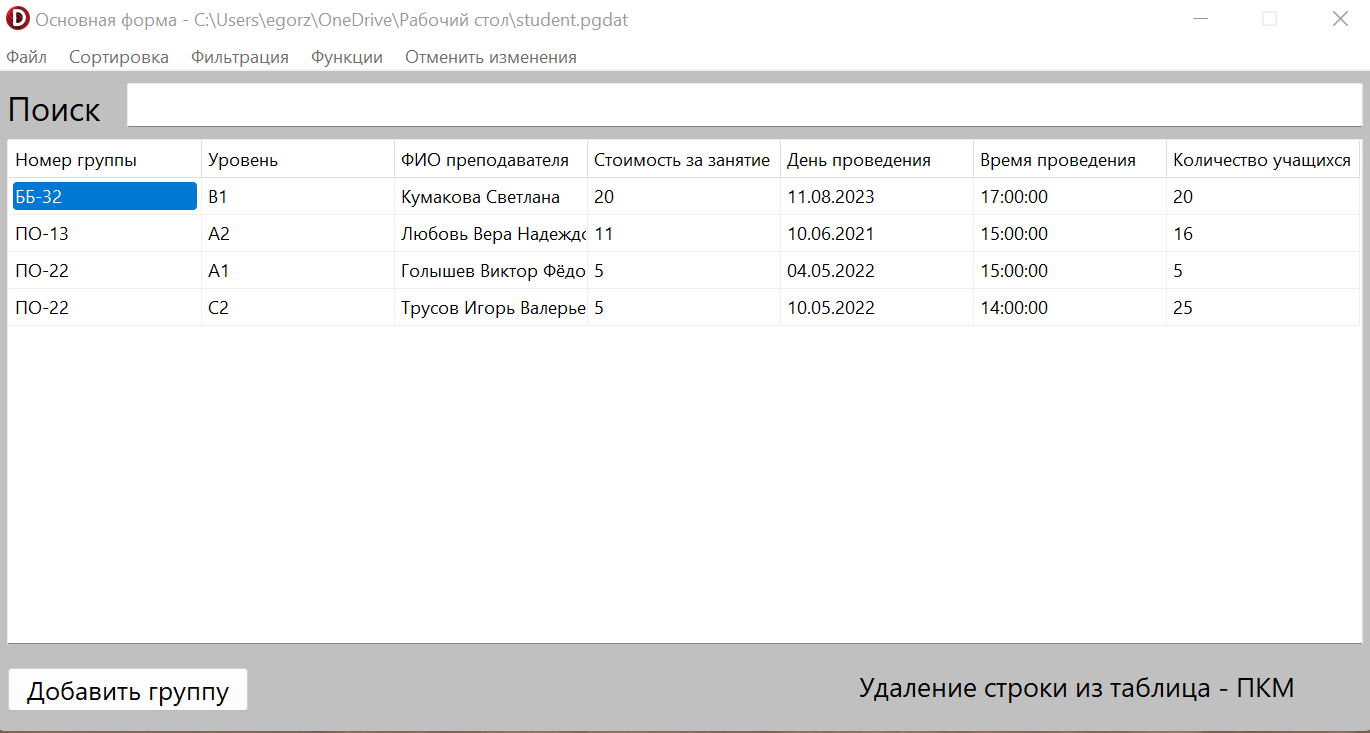


Рисунок 4.4 ⎯ Результат открытия файла

Чтобы начать работать с новой группой в таблице требуется на основной форме нажать кнопку «Добавить группу», появится новая форма «Добавить группу», где необходимо заполнить все поля и после их заполнения нажать на кнопку «Подтвердить» на новой форме, после чего программа предложит пользователю сохранить файл на компьютере.

Форма «Добавить группу» представлена на рисунке 4.5.

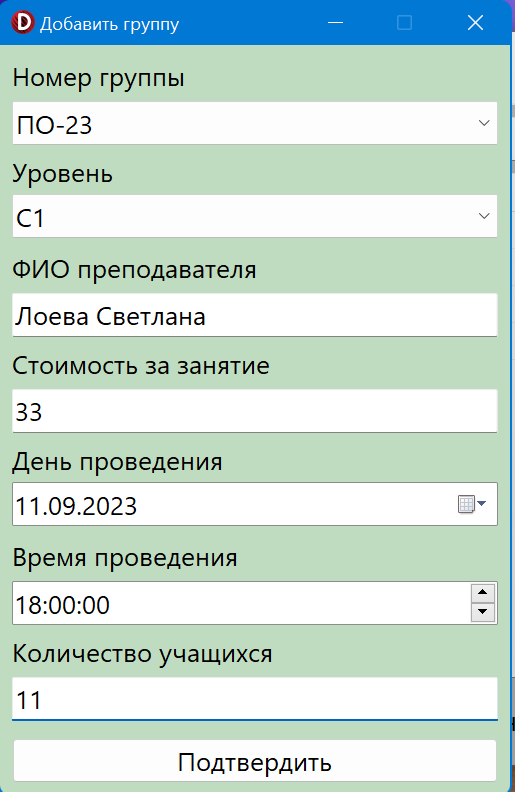


Рисунок 4.5 ⎯ Форма «Добавить группу»

Если в результате добавления записи какое-либо поле было не заполнено, то будет выведено соответствующее сообщение. На рисунке 4.6 приведено сообщение, которое появляется при не заполненных полях.

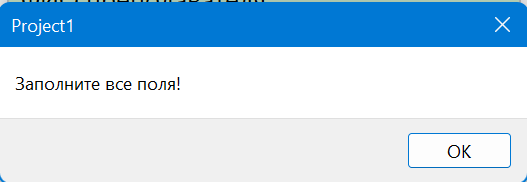


Рисунок 4.6 ⎯ Сообщение о заполнении данных

Если файл не сохранен на компьютере, то программа предложит пользователю сохранить файл при добавлении, представлено на рисунке 4.7.

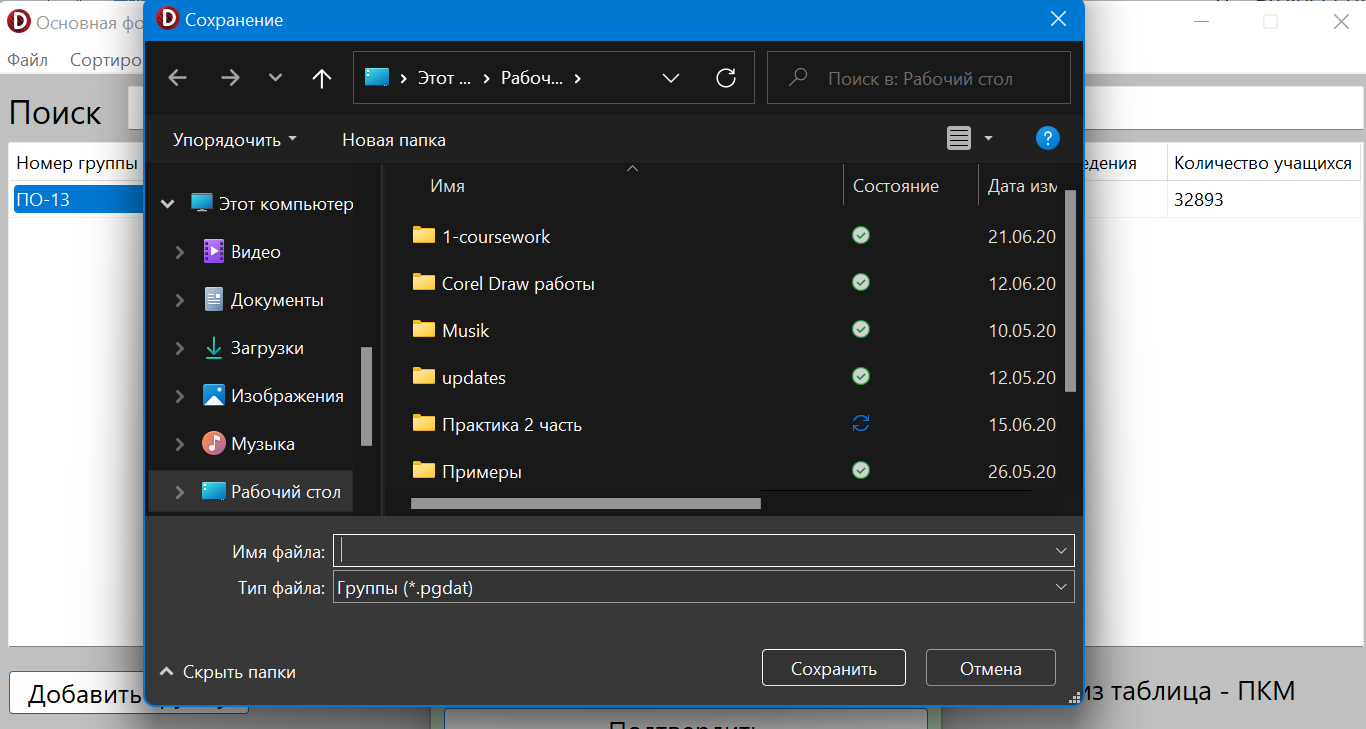


Рисунок 4.7 ⎯ Предложение сохранения файла на компьютере пользователя

На рисунке 4.8 приведен результат применения добавления группы.

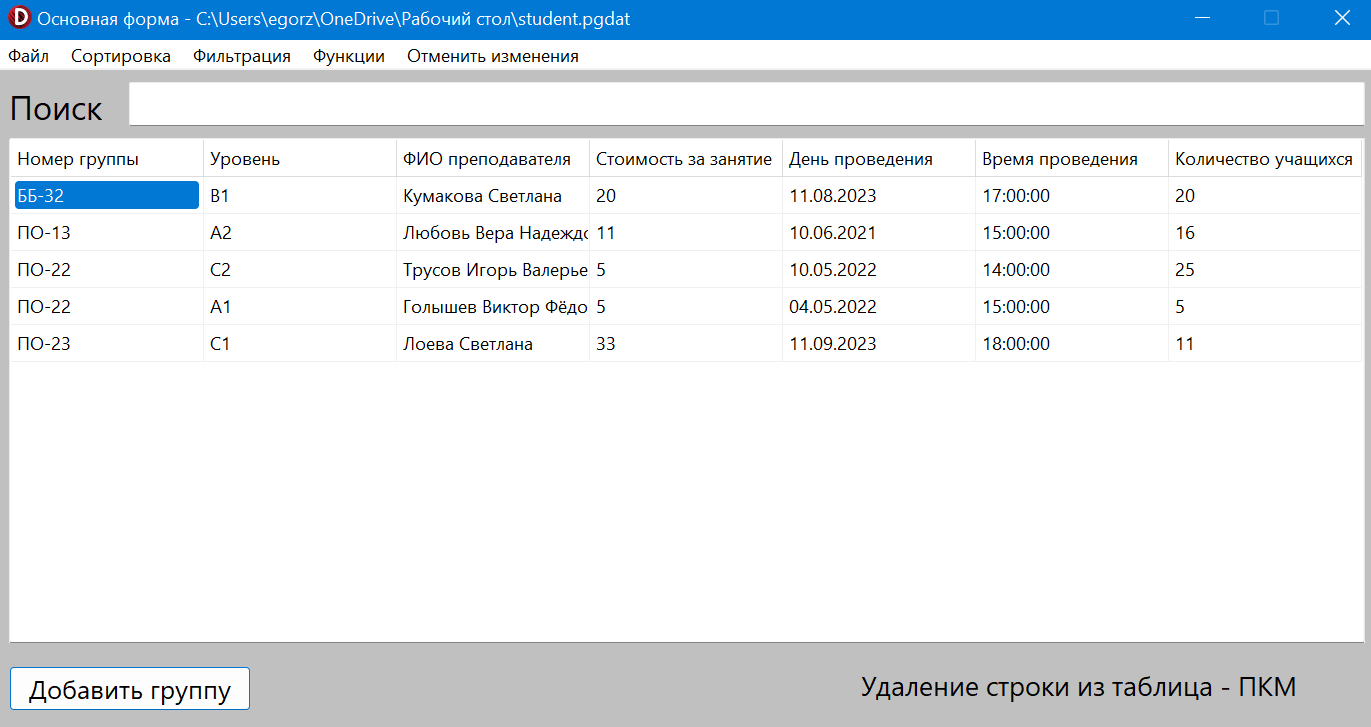


Рисунок 4.8 ⎯ Результат применения добавления группы

На рисунке 4.9 представлено редактирование атрибутов группы, которое активируется двойным кликом ЛКМ по атрибуту в таблице.

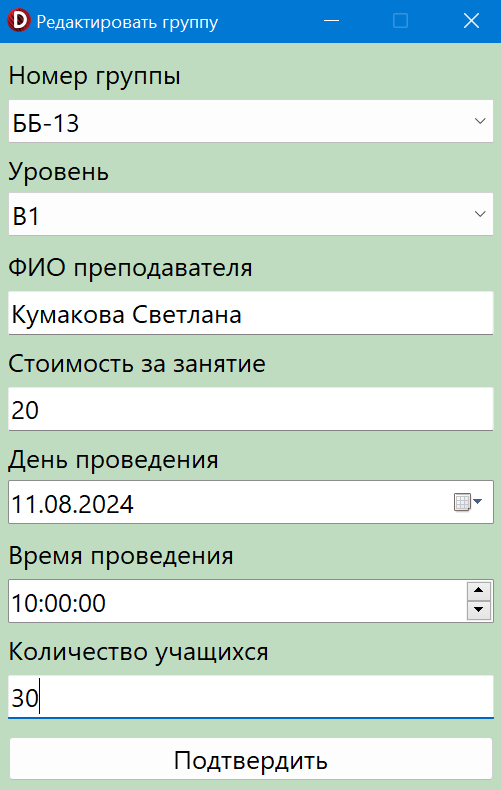


Рисунок 4.9 ⎯ Редактирование атрибутов группы

На рисунке 4.10 приведен результат применения редактирования атрибутов группы.

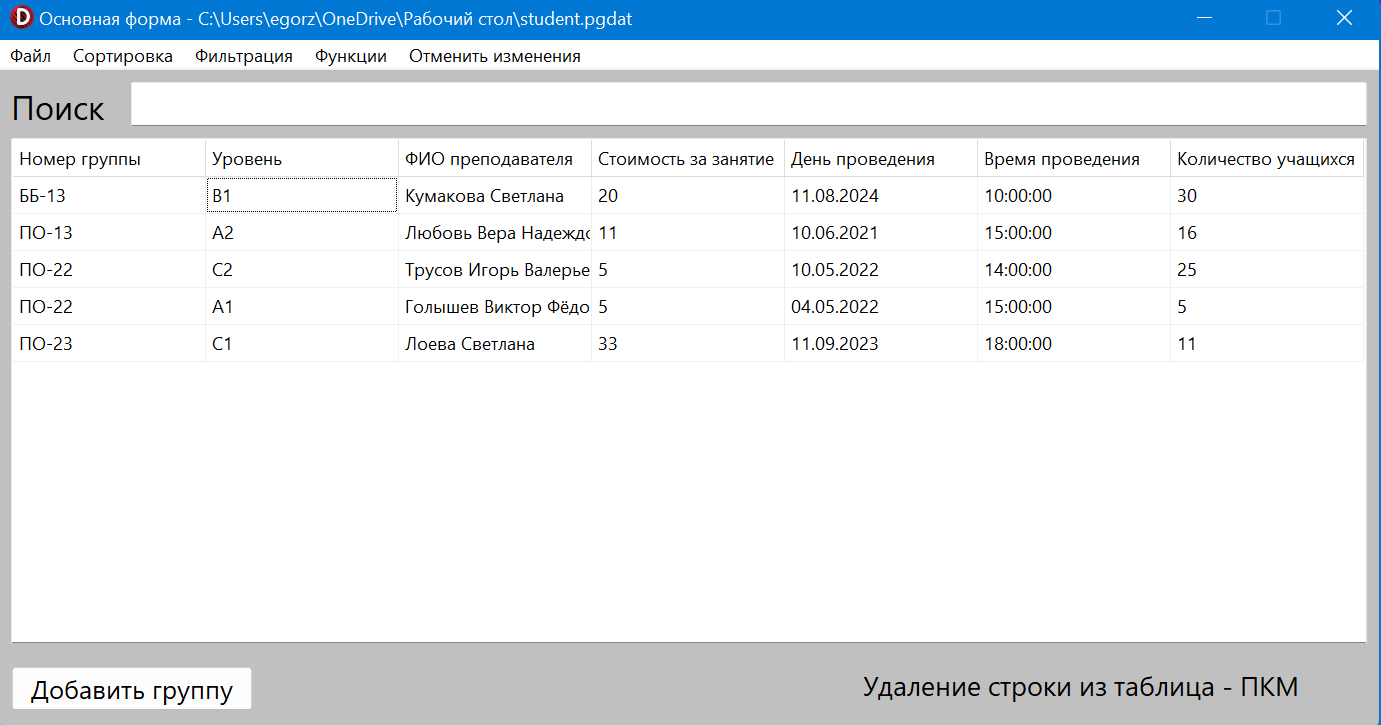


Рисунок 4.10 ⎯ Результат применения редактирования атрибутов группы

На рисунке 4.11 представлено удаление группы, которое активируется ПКМ по нужной строке таблицы с группой и в новом уведомлении программа просит подтвердить удаление.

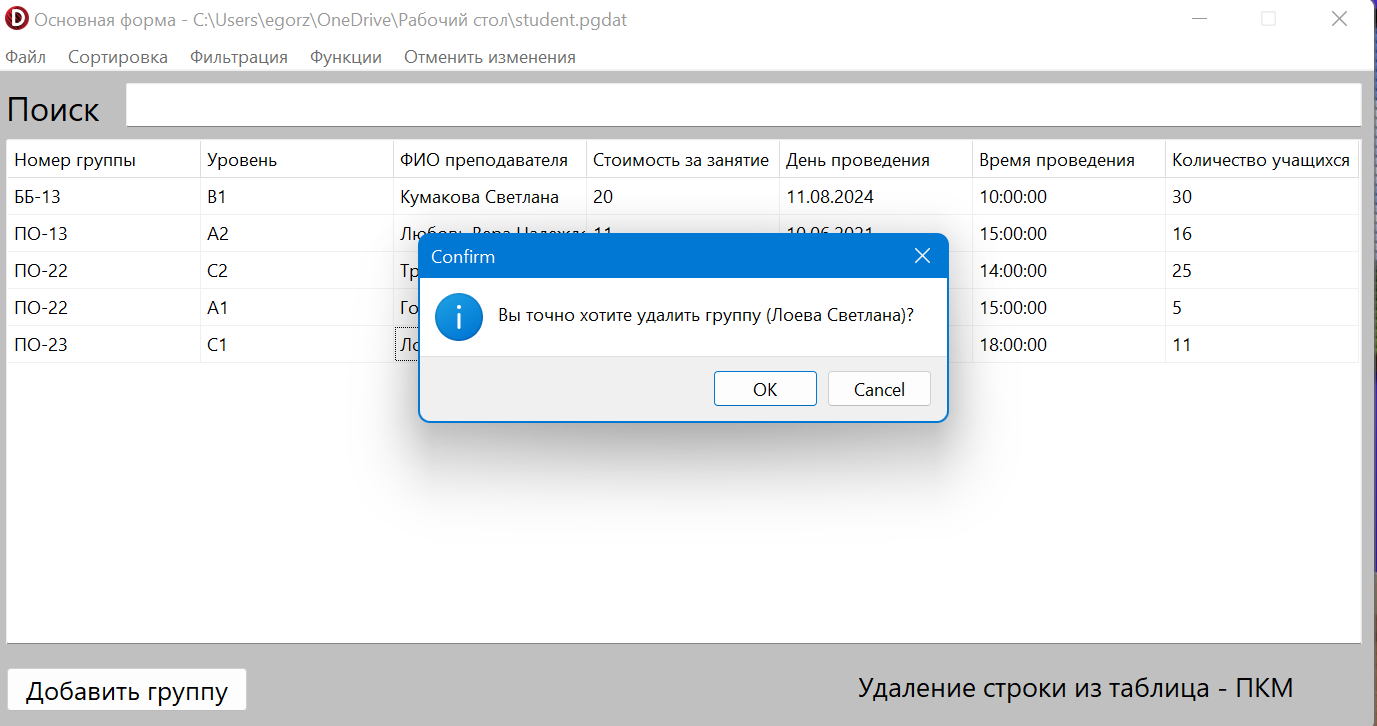


Рисунок 4.11 ⎯ Удаление группы

На рисунке 4.12 представлено результат удаления группы из таблицы.

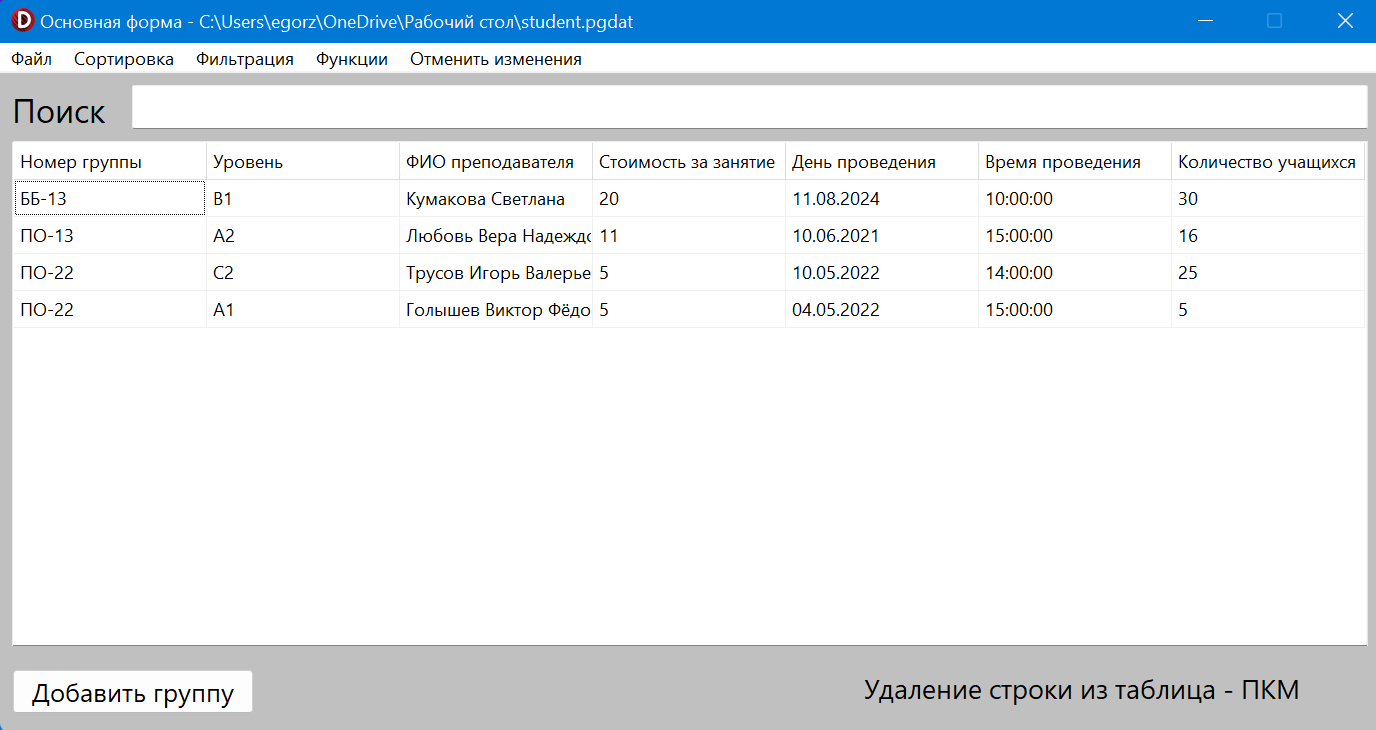


Рисунок 4.12 ⎯ Результат удаления группы из таблицы

Пользователь может осуществить сортировку содержимого, выбрав соответствующий пункт в меню программы. Сортировка осуществляется при нажатии пользователем на пункт «Сортировка» в меню программы, выбрав по какому атрибуту, по возрастанию или убыванию. Сортировка доступна по любому атрибуту групп.

На рисунке 4.13 представлено применение сортировки по количеству учащихся.

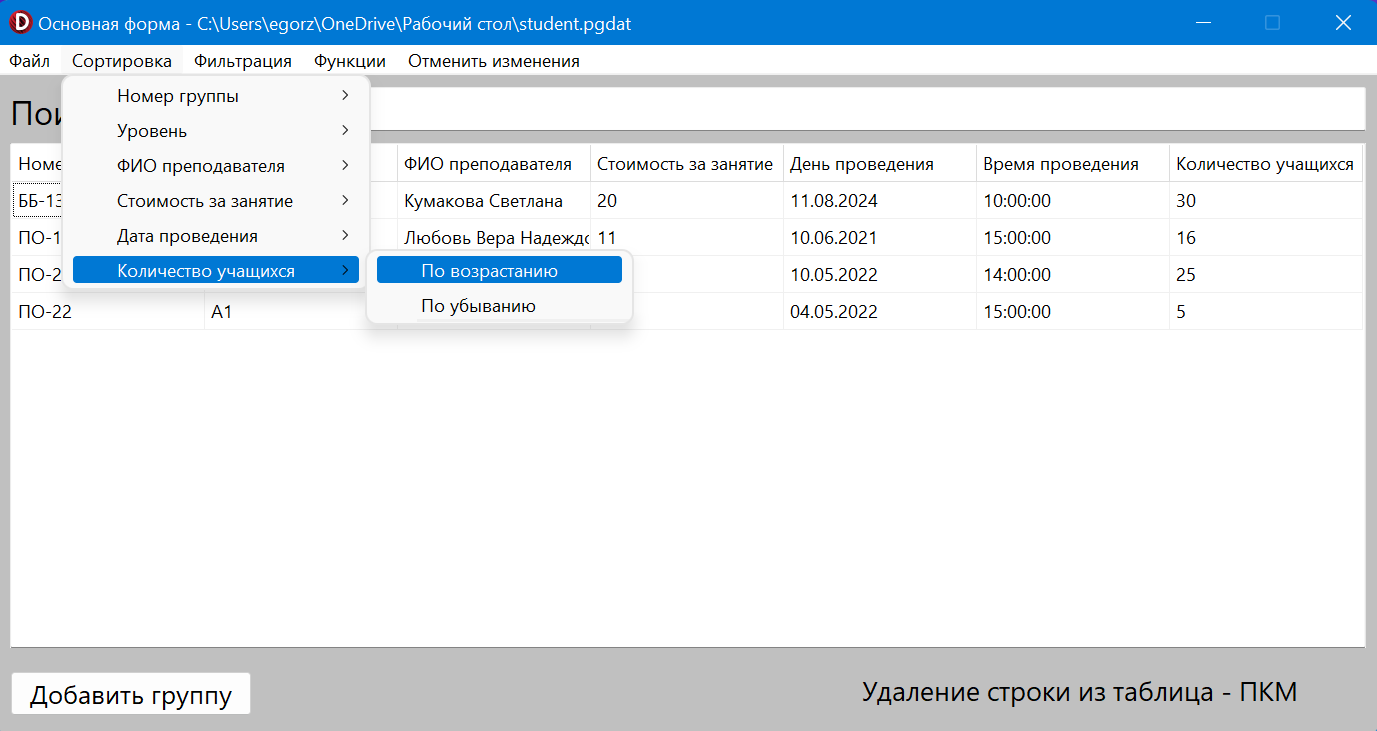


Рисунок 4.13 ⎯ Применение сортировки по количеству учащихся

На рисунке 4.14 представлен результат сортировки по атрибуту.

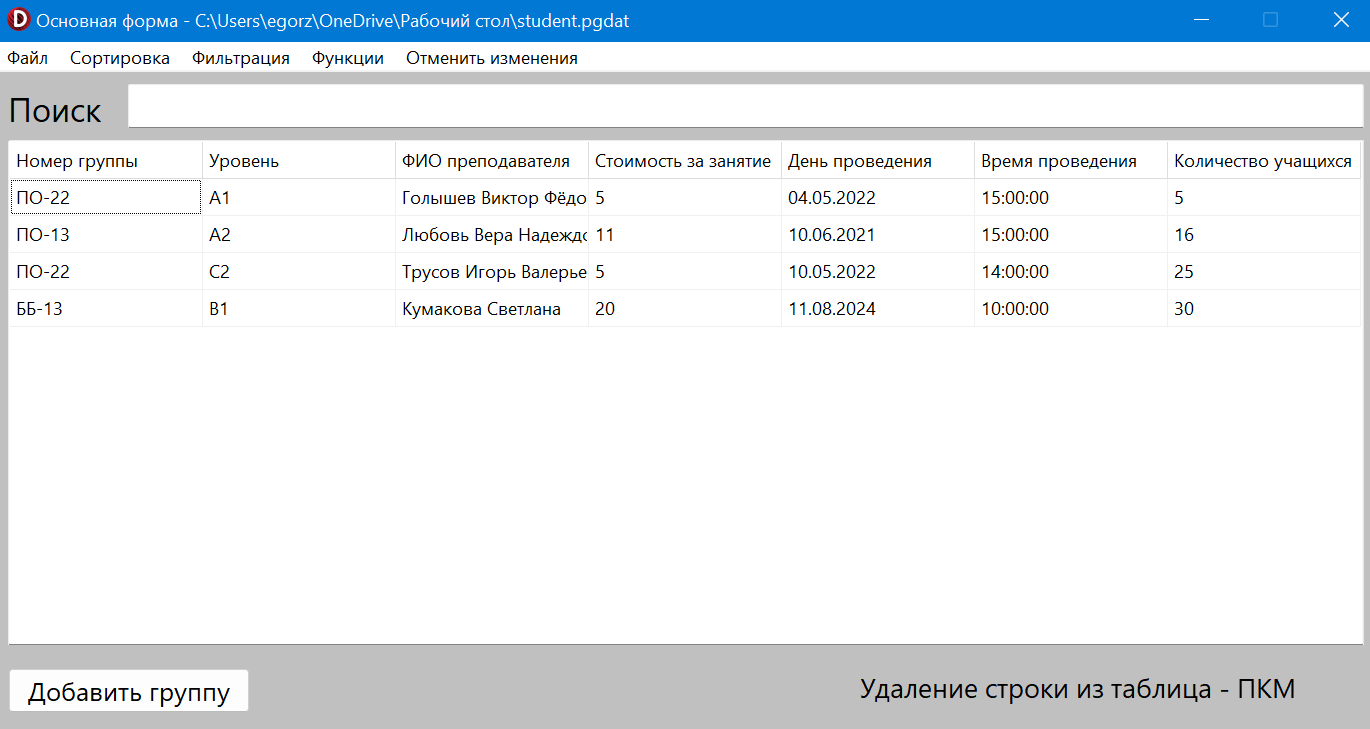


Рисунок 4.14 ⎯ Результат сортировки по атрибуту

Чтобы осуществить фильтрацию, пользователь должен нажать на пункт «Фильтрация» в меню программы и выбрать по какому атрибуту нужно отфильтровать таблицу. После нажатия появится уведомление для того, чтобы ввести фильтр.

На рисунке 4.15 представлено применение фильтра по номеру группы.

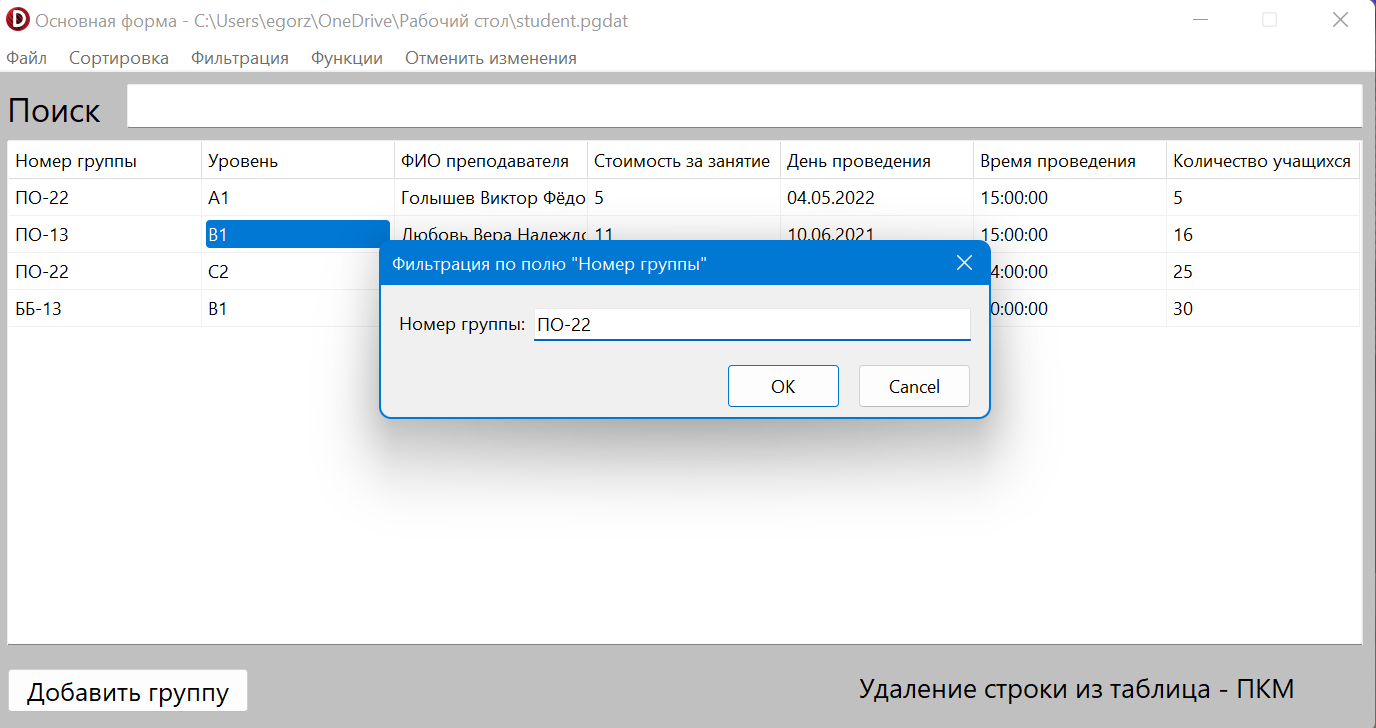


Рисунок 4.15 ⎯ Применение фильтрации по номеру группы

На рисунке 4.16 представлено применение фильтра по дате проведения.

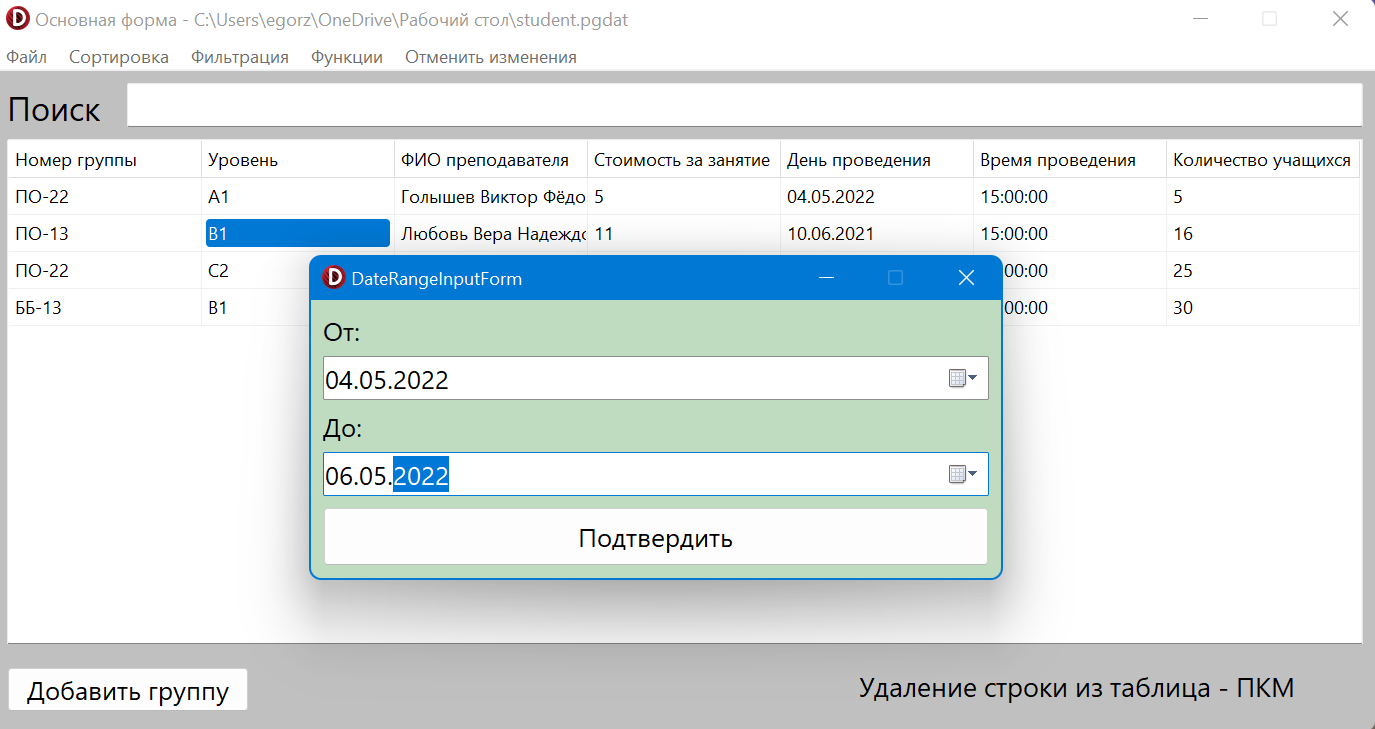


Рисунок 4.16 ⎯ Применение фильтрации по дате проведения

На рисунке 4.17 представлен результат применения фильтров.

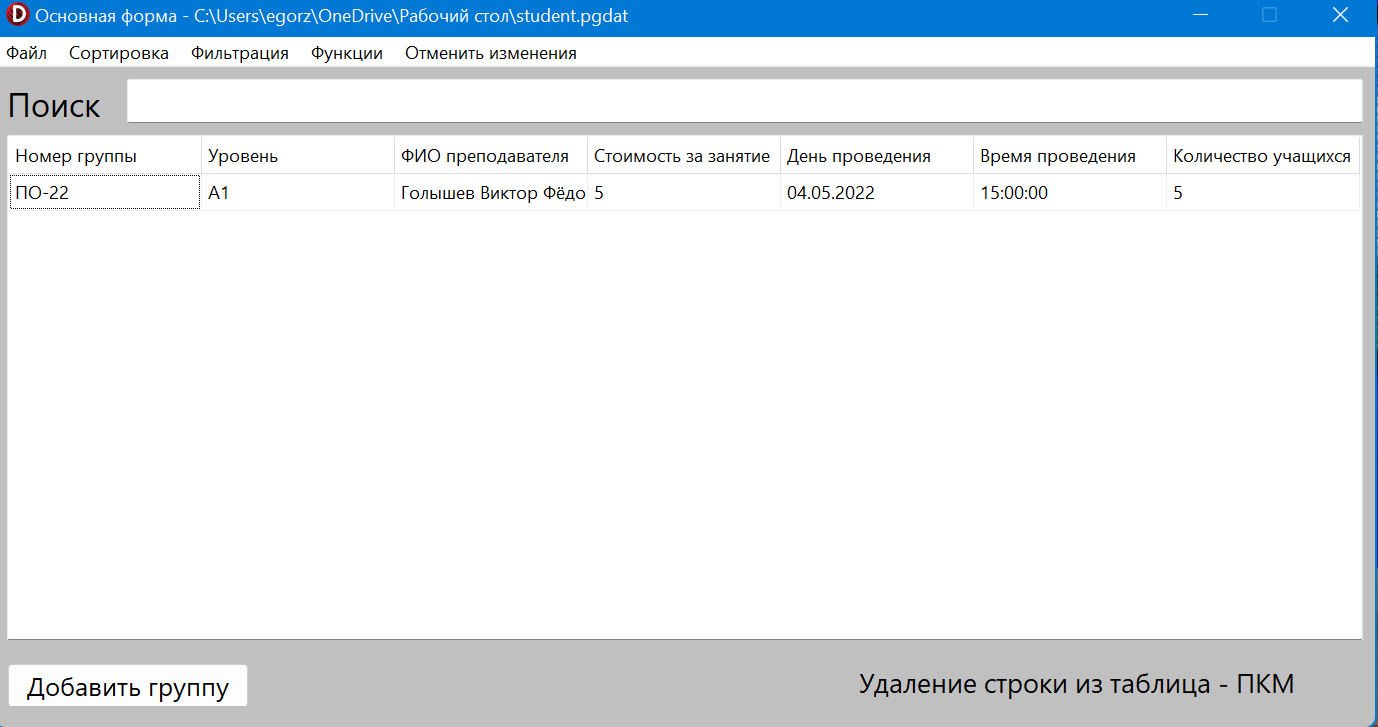


Рисунок 4.17 ⎯ Результат применения фильтров

Для применения функции «Увеличить стоимость занятий на 10% для определенного уровня языка» нужно в меню нажать на вкладку «Функции» и выбрать данную функцию. После чего появится форма, где пользователю нужно выбрать уровень языка и нажать кнопку «Применить» и изменения появятся в таблице.

На рисунке 4.18 представлено применение функции «Увеличить стоимость занятий на 10% для определенного уровня языка».

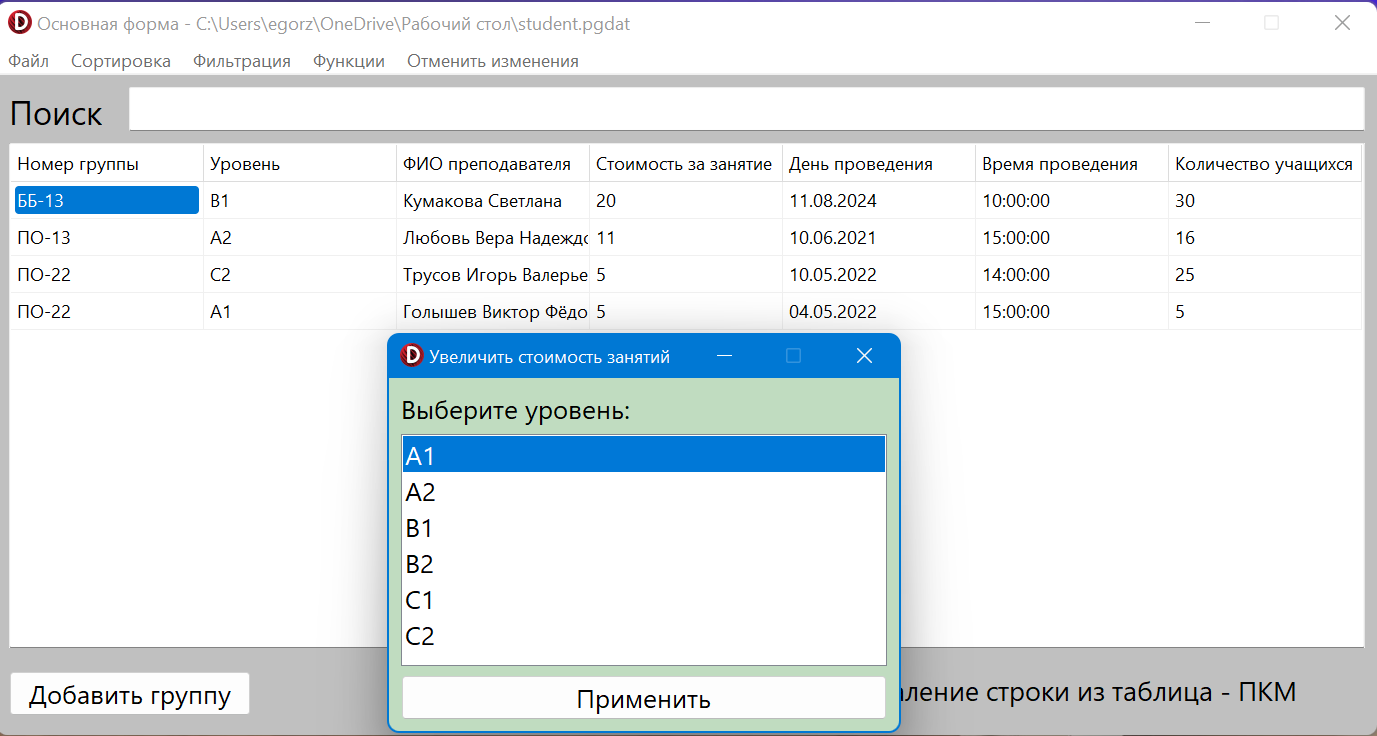


Рисунок 4.18 ⎯ Применение функции «Увеличить стоимость занятий на 10% для определенного уровня языка»

На рисунке 4.19 представлен результат применения функции «Увеличить стоимость занятий на 10% для определенного уровня языка».

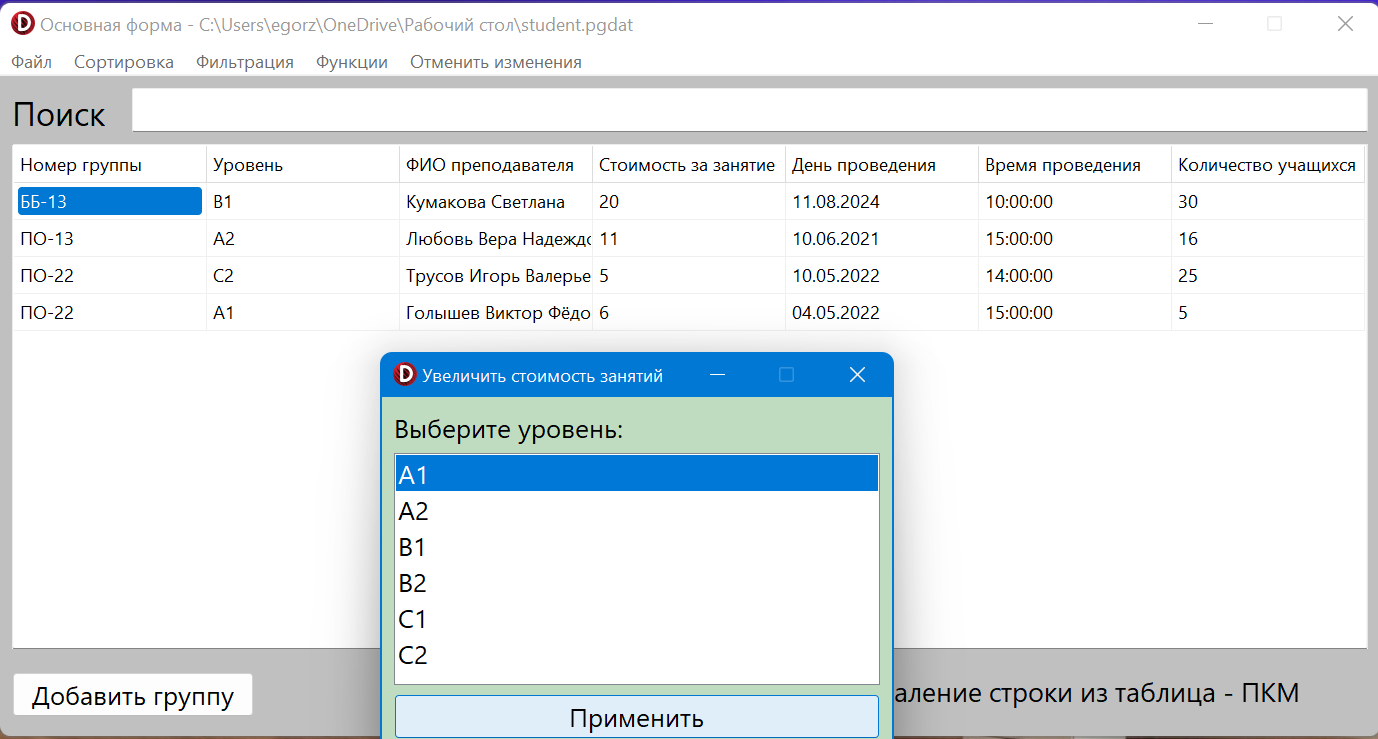


Рисунок 4.19 ⎯ Результат применения функции «Увеличить стоимость занятий на 10% для определенного уровня языка»

Для применения функции «Вывести информацию о группах, занимающихся в определенный день недели» нужно в меню нажать на вкладку «Функции» и выбрать данную функцию. После чего появится форма, где пользователю нужно выбрать день недели и нажать кнопку «Применить» и изменения появятся в таблице.

На рисунке 4.20 представлено применение функции «Вывести информацию о группах, занимающихся в определенный день недели».

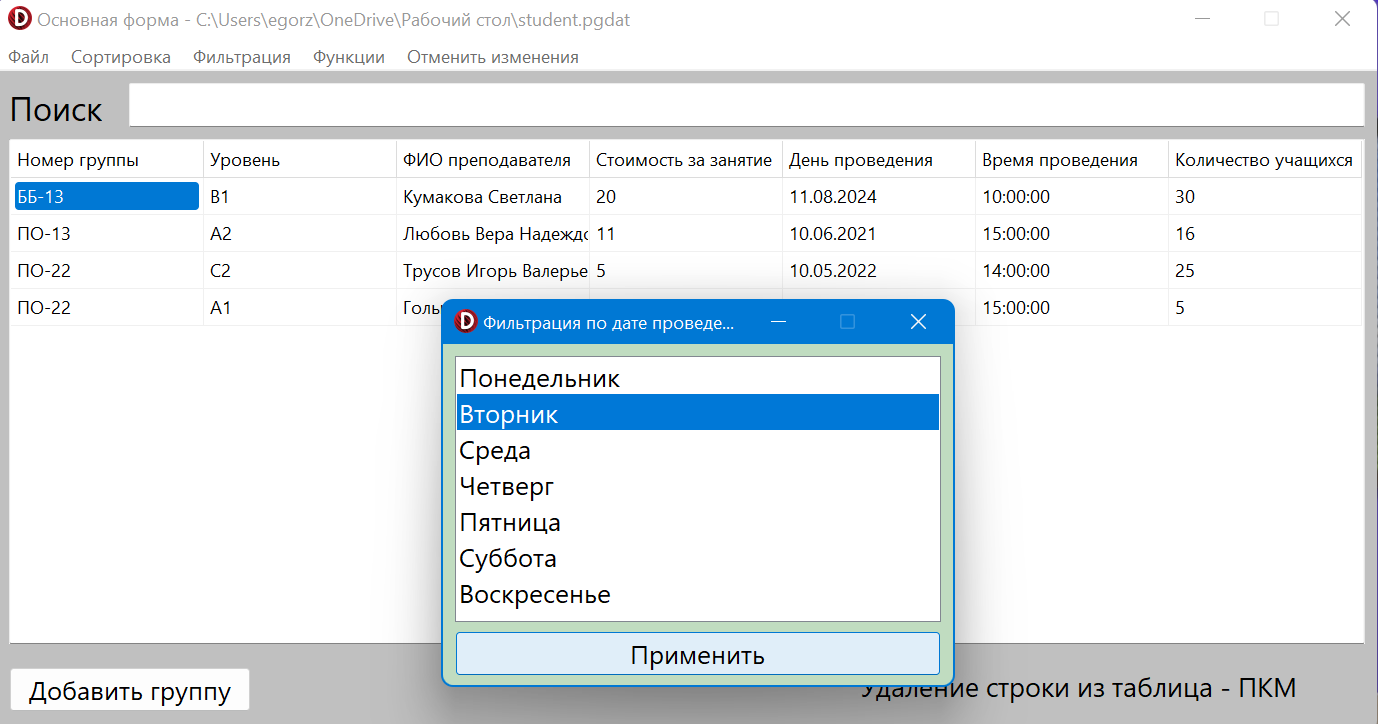


Рисунок 4.20 ⎯ Применение функции «Вывести информацию о группах, занимающихся в определенный день недели»

На рисунке 4.21 представлен результат применения функции «Вывести информацию о группах, занимающихся в определенный день недели».

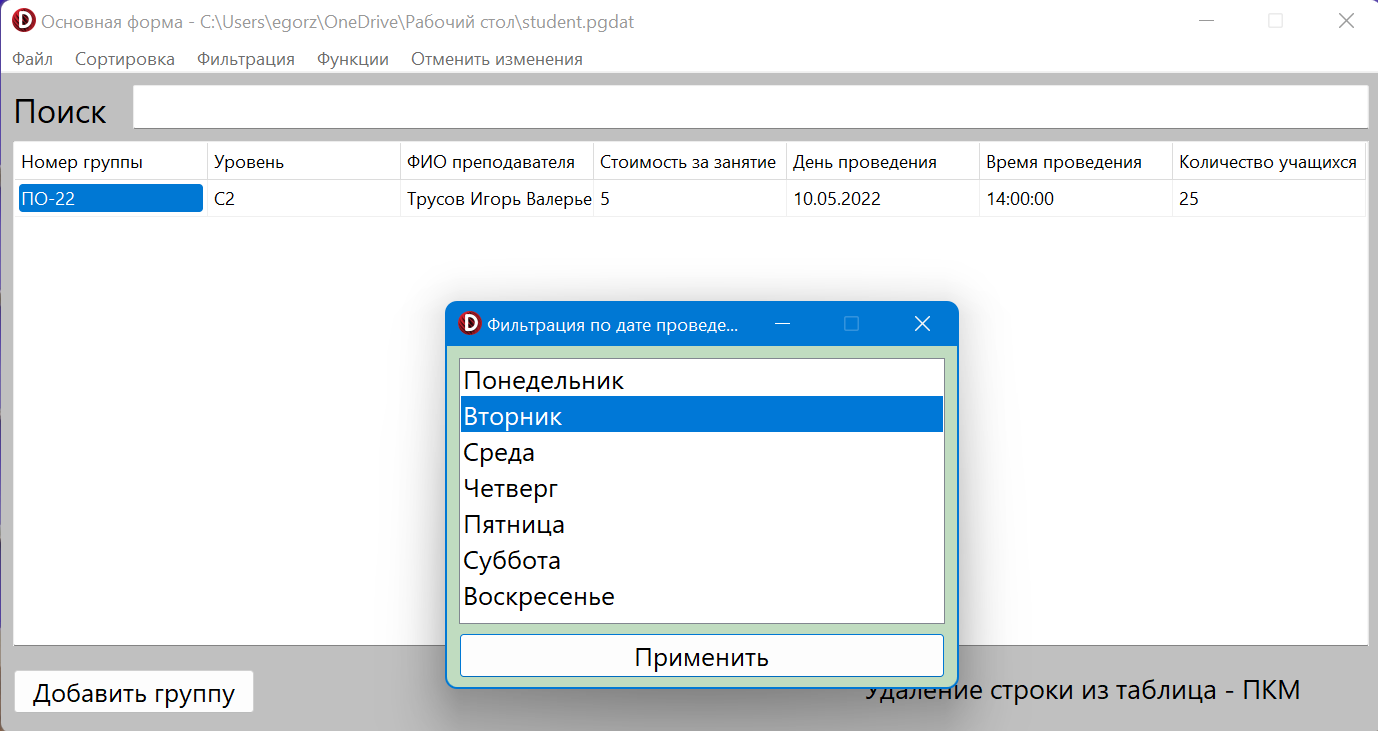


Рисунок 4.21 ⎯ Результат применения функции «Вывести информацию о группах, занимающихся в определенный день недели»

Для применения функции «Удалить группы, где меньше 10 учащихся» нужно в меню нажать на вкладку «Функции» и выбрать данную функцию после чего изменения появятся в таблице.

На рисунке 4.22 представлено применение функции «Удалить группы, где меньше 10 учащихся».

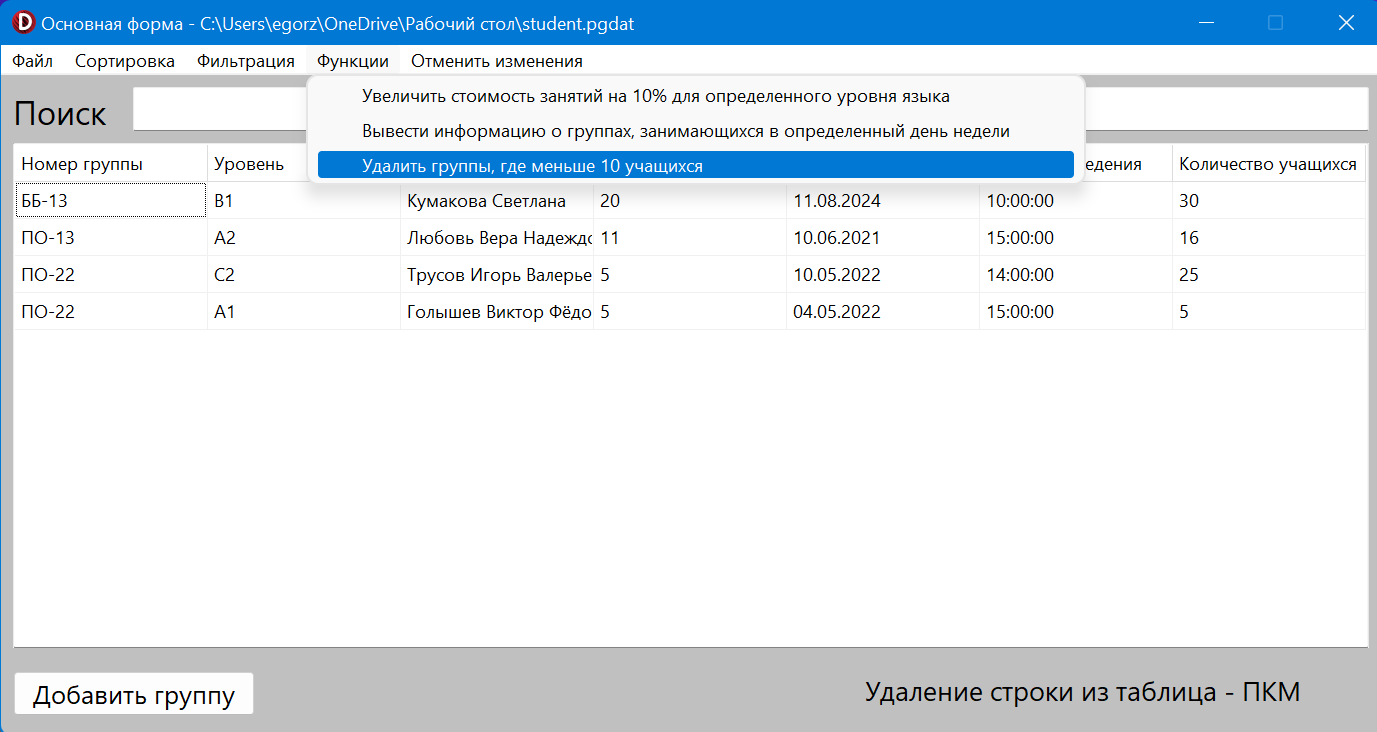


Рисунок 4.22 ⎯ Применение функции «Удалить группы, где меньше 10 учащихся»

На рисунке 4.23 представлен результат применения функции «Удалить группы, где меньше 10 учащихся».

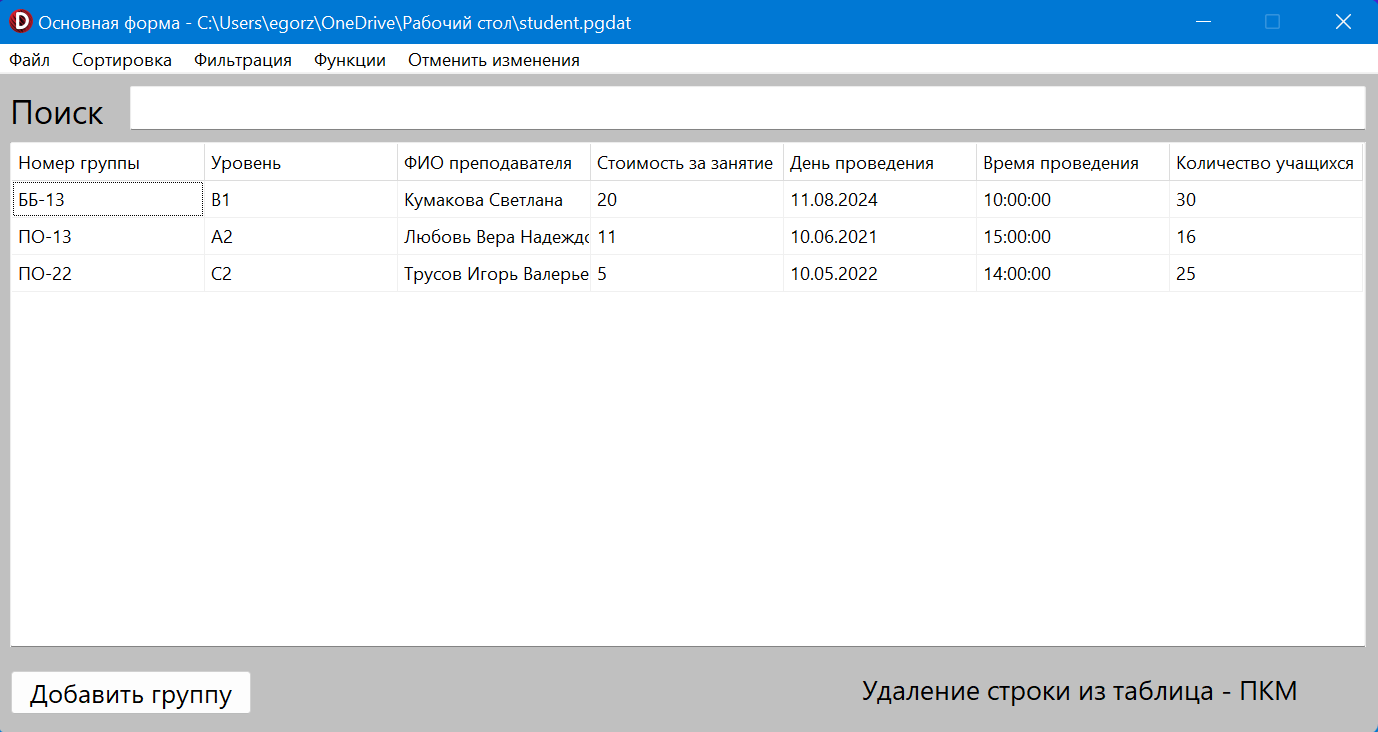


Рисунок 4.23 ⎯ Результат применения функции «Удалить группы, где меньше 10 учащихся»

Чтобы отменить изменения, которые применялись в таблице нужно в меню нажать на кнопку «Отменить изменения». Не отменяются только сохраненные изменения, добавление и удаление автоматически сохраняется.

На рисунке 4.24 представлен применение кнопки «Отменить изменения».

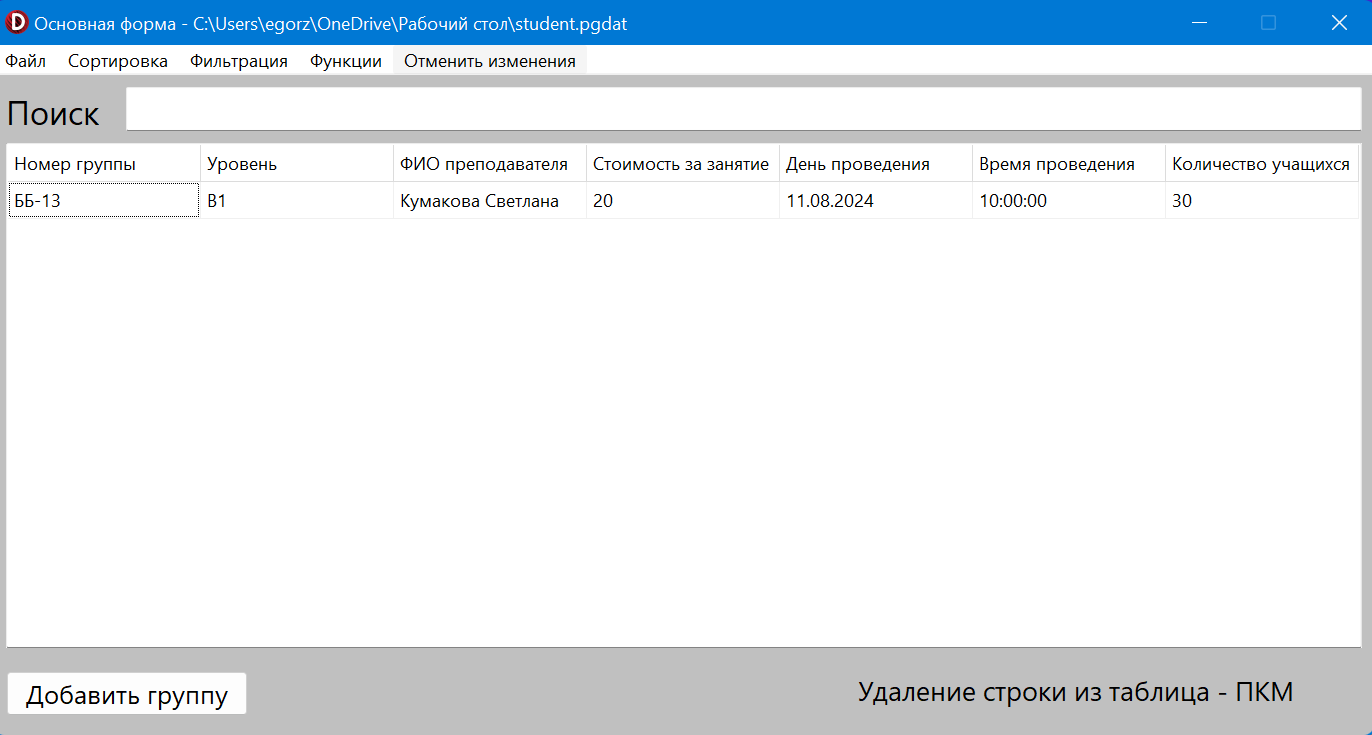


Рисунок 4.24 ⎯ Применение кнопки «Отменить изменения»

На рисунке 4.25 представлен результат применения кнопки «Отменить изменения».

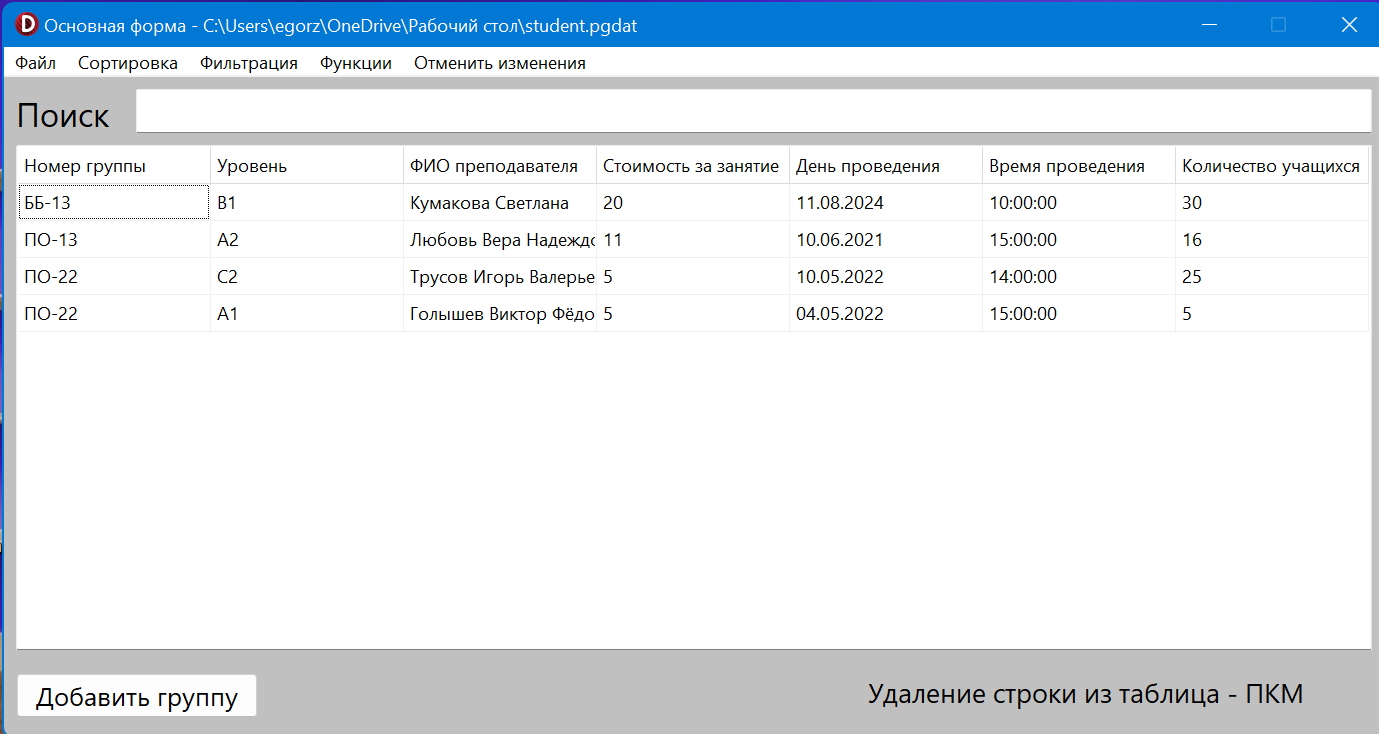


Рисунок 4.25 ⎯ Результат применения кнопки «Отменить изменения»

Чтобы воспользоваться поиском, пользователь должен ввести символы в верхнюю строку, по которым будет осуществляться поиск.

На рисунке 4.26 представлен результат символьного поиска.

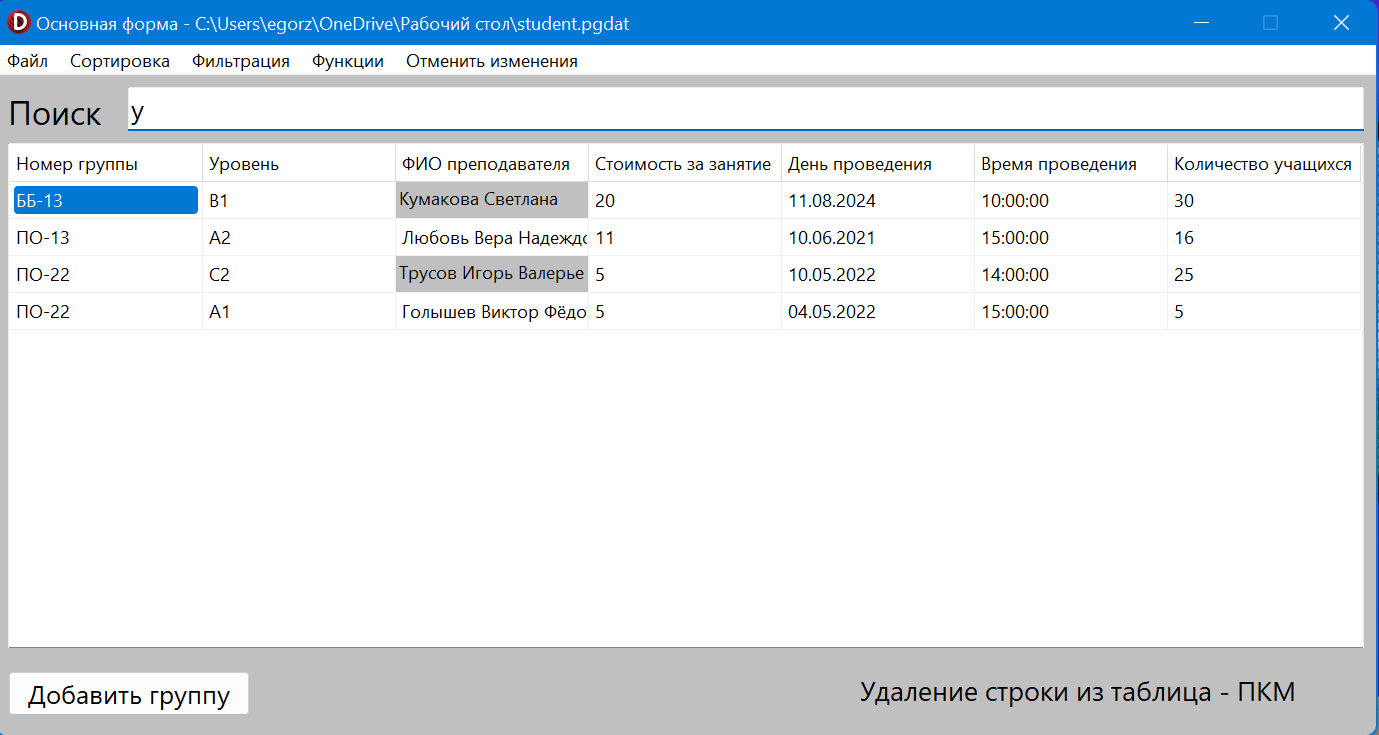


Рисунок 4.26 ⎯ Результат символьного поиска

Чтобы сохранить изменения в файле нужно в меню нажать на вкладку «Файл» и выбрать пункт «Сохранить».

Сохранение изменений в файле представлено на рисунке 4.27.

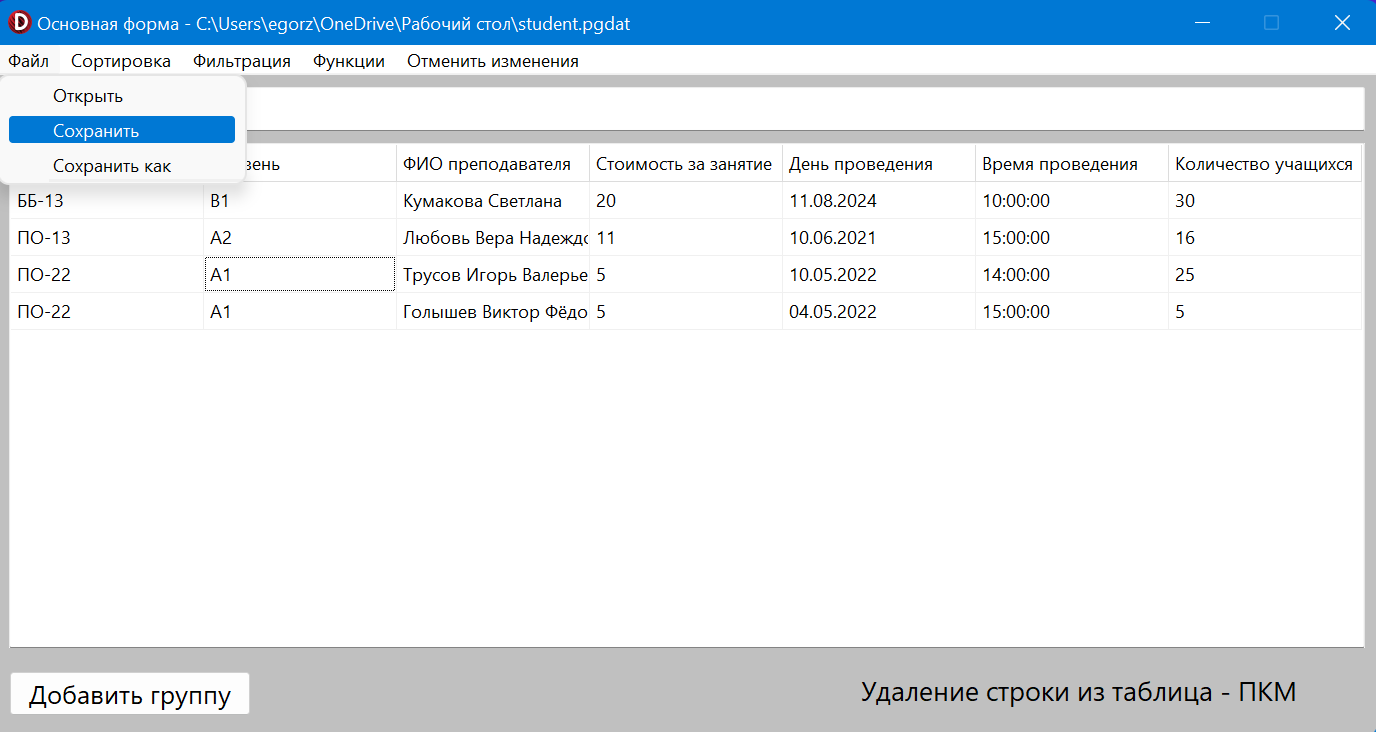


Рисунок 4.27 ⎯ Сохранение изменений в файле

Чтобы сохранить файл на компьютер нужно в меню нажать на вкладку «Файл» и выбрать пункт «Сохранить как», откроется диалоговое окно, где пользователь и произведет сохранение.

Сохранение файла на компьютер представлено на рисунке 4.28.

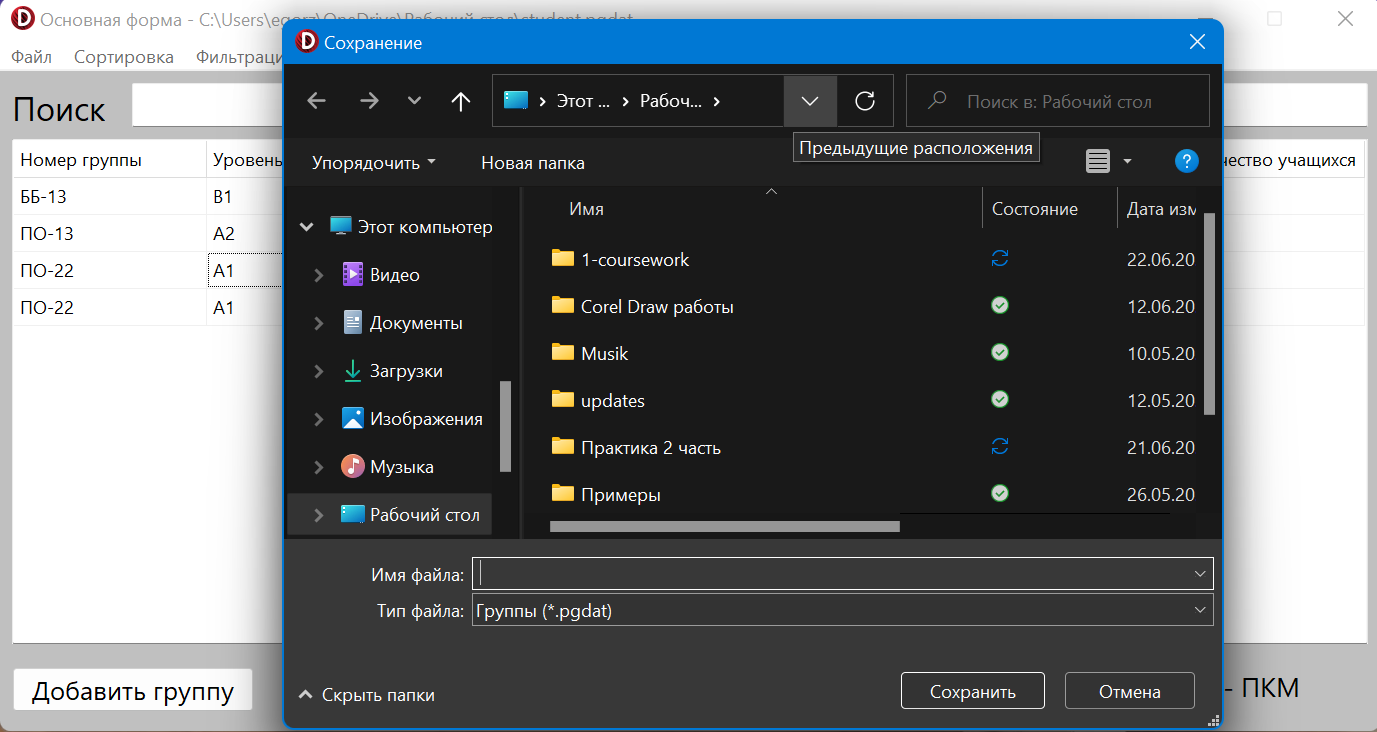


Рисунок 4.28 ⎯ Сохранение файла на компьютер

# Заключение

Исходя из цели курсового проекта, был изучен теоретический материал на тему «Подпрограммы. Процедуры и функции» и разработана информационно-поисковая система «Курсы английского языка» на языке программирования «Delphi».

Для достижения цели были поставлены и решены следующие основные задачи, которые необходимо было решить в ходе работы:

* организована информационная база проекта;
* описаны алгоритмы и технологии решения;
* разработан интерфейс программы;
* обеспечено создание нового файла;
* разработан стандартный функционал;
* разработаны функции индивидуального задания;
* организована проверка на корректность ввода;
* протестировано приложение;
* написана пояснительная записка.

Для реализации всей разработки был выбран язык программирования «Delphi», с помощью которого можно быстро и удобно разрабатывать приложения. Наличие высокоскоростного компилятора обеспечивает быстрый и беспроблемный перевод программ в машинный код. Визуальное построение приложений способствует быстрому и качественному созданию интерфейса программы.

Использование информационно-поисковой системы «Курсы английского языка» упрощает работу пользователя с информацией о преподавателях, учениках и курсах. Система используется для оптимизации учебного процесса.

**Список использованных источников**

1. Хомоненко А. Д. Delphi 7: учебный курс/ А. Д. Хомоненко, В. Э. Гофман, С. И. Бобровский Пб.: БХВ-Петербург, 2008. — 576с.
2. Федотова С. В. Курс Delphi для начинающих. Полигон нестандартных задач / Федотова С. В. СПб.: БХВ-Петербург, 2007. — 360 с.
3. Культин Н. Б. Основы программирования в Delphi / Н. Б. Культин, С. М. Окулов — Мн.: БГУ, 2008. — 480 с.
4. Бабушкина И. А. Практикум по объектно-ориентированному программированию/ Бабушкина И. А — 5-е изд. электрон. — М.: Лаборатория знаний, 2020. — 369 с.
5. Осипов А.В. [PascalABC.NET: введение в программирование](http://pascalabc.net/downloads/OsipovBook/%D0%9A%D0%BD%D0%B8%D0%B3%D0%B0%D0%94%D0%BB%D1%8F%D0%A1%D0%B0%D0%B9%D1%82%D0%B0.pdf)/ А. В. Осипов — Ростов-на-Дону, 2019. — 572с.
6. Борисов С. В. Введение в среду визуального программирования Turbo Delphi/ С. В. Борисов, С. С. Комалов, И. Л. Серебрякова — Учеб. Пособие — Ч.1. — М.: Изд-во МГТУ им. И. М. Баумена 2008 — 80 с.
7. Тюкачев Н. А. Программирование графики в делфи / Н. А. Тюкачев, И. В. Илларионов, В. Г. Хлебостроев — СПб.: БХВ-Петербург, 2008. — 784 с.

# ПРИЛОЖЕНИЕ А

(обязательное)

**Листинг программы**

Текст модуля «Unit1»:

unit Unit1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type

TForm1 = class(TForm)

HeaderLabel: TLabel;

AuthorLabel: TLabel;

Button1: TButton;

procedure FormActivate(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

uses Unit2;

{$R \*.dfm}

procedure TForm1.Button1Click(Sender: TObject);

begin

Form2.Show;

Form1.Hide;

end;

procedure TForm1.FormActivate(Sender: TObject);

begin

HeaderLabel.Left := round((Form1.ClientWidth - HeaderLabel.Width) / 2);

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

Form1.color:= $C0C0C0;

end;

end.

Текст модуля «Unit2»:

unit Unit2;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Vcl.Grids, Vcl.StdCtrls, DateUtils,

System.Generics.Collections, System.Generics.Defaults;

type

TForm2 = class(TForm)

MainMenu1: TMainMenu;

N1: TMenuItem;

OpenMenuItem: TMenuItem;

SaveMenuItem: TMenuItem;

SaveAsMenuItem: TMenuItem;

StringGrid1: TStringGrid;

OpenDialog1: TOpenDialog;

AddGroupButton: TButton;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N6: TMenuItem;

N7: TMenuItem;

N9: TMenuItem;

NumberSortAsc: TMenuItem;

NumberSortDesc: TMenuItem;

DateSortAsc: TMenuItem;

DateSortDesc: TMenuItem;

LevelSortAsc: TMenuItem;

LevelSortDesc: TMenuItem;

FullnameSortAsc: TMenuItem;

FullnameSortDesc: TMenuItem;

PriceSortAsc: TMenuItem;

PriceSortDesc: TMenuItem;

StudentsCountSortAsc: TMenuItem;

StudentsCountSortDesc: TMenuItem;

N8: TMenuItem;

N10: TMenuItem;

IncreasePriceMenuItem: TMenuItem;

SearchEdit: TEdit;

SearchLabel: TLabel;

DumpGroupsInDayMenuItem: TMenuItem;

NumberFilterMenuItem: TMenuItem;

LevelFilterMenuItem: TMenuItem;

FullnameFilterMenuItem: TMenuItem;

PriceFilterMenuItem: TMenuItem;

DateFilterMenuItem: TMenuItem;

StudentsCountFilter: TMenuItem;

SaveDialog1: TSaveDialog;

HelpLabelDelete: TLabel;

N11: TMenuItem;

N101: TMenuItem;

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormActivate(Sender: TObject);

procedure SaveAsMenuItemClick(Sender: TObject);

procedure AddGroupButtonClick(Sender: TObject);

procedure OpenMenuItemClick(Sender: TObject);

procedure SaveMenuItemClick(Sender: TObject);

procedure NumberSortAscClick(Sender: TObject);

procedure NumberSortDescClick(Sender: TObject);

procedure DateSortAscClick(Sender: TObject);

procedure DateSortDescClick(Sender: TObject);

procedure LevelSortAscClick(Sender: TObject);

procedure LevelSortDescClick(Sender: TObject);

procedure FullnameSortAscClick(Sender: TObject);

procedure FullnameSortDescClick(Sender: TObject);

procedure PriceSortAscClick(Sender: TObject);

procedure PriceSortDescClick(Sender: TObject);

procedure StudentsCountSortAscClick(Sender: TObject);

procedure StudentsCountSortDescClick(Sender: TObject);

procedure DateFilterClick(Sender: TObject);

procedure DeleteGroupsButtonClick(Sender: TObject);

procedure IncreasePriceMenuItemClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure SearchEditChange(Sender: TObject);

procedure StringGrid1ContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

procedure DumpGroupsInDayMenuItemClick(Sender: TObject);

procedure NumberFilterMenuItemClick(Sender: TObject);

procedure LevelFilterMenuItemClick(Sender: TObject);

procedure FullnameFilterMenuItemClick(Sender: TObject);

procedure PriceFilterMenuItemClick(Sender: TObject);

procedure DateFilterMenuItemClick(Sender: TObject);

procedure StudentsCountFilterClick(Sender: TObject);

procedure StringGrid1DblClick(Sender: TObject);

procedure N11Click(Sender: TObject);

procedure N101Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

groupRecord = record

number, level, fullName, price, day, time, studentsCount: string[20];

end;

procedure UpdateStringGridFromFile(path: string);

function createGroupFromStringGrid(rowIndex: integer): groupRecord;

const fields: array of String = ['Номер группы', 'Уровень', 'ФИО преподавателя', 'Стоимость за занятие', 'День проведения', 'Время проведения', 'Количество учащихся'];

const levels: array of String = ['A1', 'A2', 'B1', 'B2', 'C1', 'C2'];

var

Form2: TForm2;

storageFile: file of groupRecord;

storageFilePath: string;

i, j: integer;

implementation

uses Unit1, Unit3, Unit4, Unit5, Unit6;

{$R \*.dfm}

function createGroupFromStringGrid(rowIndex: integer): groupRecord;

var group: groupRecord;

begin

group.number := Form2.StringGrid1.Cells[0, rowIndex];

group.level := Form2.StringGrid1.Cells[1, rowIndex];

group.fullName := Form2.StringGrid1.Cells[2, rowIndex];

group.price := Form2.StringGrid1.Cells[3, rowIndex];

group.day := Form2.StringGrid1.Cells[4, rowIndex];

group.time := Form2.StringGrid1.Cells[5, rowIndex];

group.studentsCount := Form2.StringGrid1.Cells[6, rowIndex];

result := group;

end;

procedure SetStorageFilePath(path: string);

begin

storageFilePath := path;

Form2.Caption := 'Основная форма - ' + storageFilePath;

end;

procedure ReverseStringGrid();

var

i, j, k, cnt: integer;

s: string;

begin

cnt := Form2.StringGrid1.RowCount;

k := Form2.StringGrid1.RowCount div 2;

for j := 0 to 6 do

for i := 1 to k do

begin

s := Form2.StringGrid1.Cells[j,i];

Form2.StringGrid1.Cells[j,i] := Form2.StringGrid1.Cells[j,cnt-i];

Form2.StringGrid1.Cells[j,cnt-i] := s;

end;

end;

procedure UpdateStringGridFromFile(path: string);

var group: groupRecord;

begin

Form2.StringGrid1.RowCount := 1;

AssignFile(storageFile, path);

Reset(storageFile);

for i := 1 to Filesize(storageFile) do

begin

read(storageFile, group);

AddGroupToStringGrid(group);

end;

CloseFile(storageFile);

end;

procedure TForm2.AddGroupButtonClick(Sender: TObject);

begin

Form3.Caption := 'Добавить группу';

Form3.ShowModal;

end;

procedure TForm2.DateFilterClick(Sender: TObject);

begin

Form4.ShowModal;

end;

procedure TForm2.DateFilterMenuItemClick(Sender: TObject);

var

dateStart, dateEnd: integer;

groups: array of groupRecord;

begin

if length(storageFilePath) = 0 then

begin

ShowMessage('Данные не найдены.');

exit;

end;

DateRangeInputForm.ShowModal;

dateStart := DateTimeToUnix(DateRangeInputForm.DateTimePicker1.Date);

dateEnd := DateTimeToUnix(DateRangeInputForm.DateTimePicker2.Date);

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, Form2.StringGrid1.RowCount - 1);

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

Form2.StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if (DateTimeToUnix(StrToDateTime(groups[i].day)) >= dateStart) and (DateTimeToUnix(StrToDateTime(groups[i].day)) <= dateEnd) then AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.DateSortAscClick(Sender: TObject);

var

groups: array [1..100] of groupRecord;

group: groupRecord;

i\_min, min, groupsLength: integer;

begin

if length(storageFilePath) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

groupsLength := StringGrid1.RowCount - 1;

for i := 1 to groupsLength do

groups[i] := createGroupFromStringGrid(i);

for var i := 1 to groupsLength - 1 do

begin

min := DateTimeToUnix(StrToDateTime(groups[i].day + ' ' + groups[i].time)); i\_min := i;

for var j := i + 1 to groupsLength do

if DateTimeToUnix(StrToDateTime(groups[j].day + ' ' + groups[j].time)) < min then

begin

min := DateTimeToUnix(StrToDateTime(groups[j].day + ' ' + groups[j].time));

i\_min := j;

end;

group := groups[i\_min];

groups[i\_min] := groups[i];

groups[i] := group;

end;

StringGrid1.RowCount := 1;

for i := 1 to groupsLength do

AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.DateSortDescClick(Sender: TObject);

begin

DateSortAscClick(DateSortAsc);

ReverseStringGrid();

end;

procedure TForm2.DeleteGroupsButtonClick(Sender: TObject);

var

groups: array of groupRecord;

group: groupRecord;

begin

if length(storageFilePath) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, StringGrid1.RowCount - 1);

// Skip first row

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if StrToInt(groups[i].studentsCount) >= 10 then AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.DumpGroupsInDayMenuItemClick(Sender: TObject);

begin

Form4.ShowModal;

end;

procedure TForm2.FormActivate(Sender: TObject);

begin

StringGrid1.DefaultColWidth := round(Form2.ClientWidth / Length(fields)) - 5;

StringGrid1.ColCount := Length(fields);

for i := 0 to Length(fields) - 1 do

StringGrid1.Cells[i, 0] := fields[i];

end;

procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Form1.Close;

end;

procedure TForm2.FormCreate(Sender: TObject);

begin

Form2.Color:= $C0C0C0;

end;

procedure TForm2.FullnameFilterMenuItemClick(Sender: TObject);

var

value: string;

groups: array of groupRecord;

begin

if length(storageFilePath) = 0 then

begin

ShowMessage('Данные не найдены.');

exit;

end;

value := InputBox('Фильтрация по полю "ФИО преподавателя"', 'ФИО преподавателя:', '');

if length(value) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, Form2.StringGrid1.RowCount - 1);

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

Form2.StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if AnsiLowerCase(Trim(groups[i].fullName)) = AnsiLowerCase(Trim(value)) then AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.FullnameSortAscClick(Sender: TObject);

var

groups: array of groupRecord;

group: groupRecord;

begin

if length(storageFilePath) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, StringGrid1.RowCount - 1);

// Skip first row

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

TArray.Sort<groupRecord>(groups, TDelegatedComparer<groupRecord>.Construct(

function(const Left, Right: groupRecord): integer

begin

Result := TComparer<string>.Default.Compare(left.fullName, right.fullName);

end

));

StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.FullnameSortDescClick(Sender: TObject);

begin

FullnameSortAscClick(FullnameSortAsc);

ReverseStringGrid();

end;

procedure TForm2.IncreasePriceMenuItemClick(Sender: TObject);

begin

Form5.ShowModal;

end;

procedure TForm2.LevelFilterMenuItemClick(Sender: TObject);

var

value: string;

groups: array of groupRecord;

begin

if length(storageFilePath) = 0 then

begin

ShowMessage('Данные не найдены.');

exit;

end;

value := InputBox('Фильтрация по полю "Уровень"', 'Уровень:', '');

if length(value) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, Form2.StringGrid1.RowCount - 1);

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

Form2.StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if AnsiLowerCase(Trim(groups[i].level)) = AnsiLowerCase(Trim(value)) then AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.LevelSortAscClick(Sender: TObject);

var

groups: array of groupRecord;

group: groupRecord;

begin

if length(storageFilePath) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, StringGrid1.RowCount - 1);

// Skip first row

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

TArray.Sort<groupRecord>(groups, TDelegatedComparer<groupRecord>.Construct(

function(const Left, Right: groupRecord): integer

begin

Result := TComparer<string>.Default.Compare(left.level, right.level);

end

));

StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.LevelSortDescClick(Sender: TObject);

begin

LevelSortAscClick(LevelSortAsc);

ReverseStringGrid();

end;

procedure TForm2.N101Click(Sender: TObject);

var

groups: array of groupRecord;

group: groupRecord;

begin

if length(storageFilePath) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, StringGrid1.RowCount - 1);

// Skip first row

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if StrToInt(groups[i].studentsCount) >= 10 then AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.N11Click(Sender: TObject);

begin

UpdateStringGridFromFile(OpenDialog1.FileName);

end;

procedure TForm2.NumberFilterMenuItemClick(Sender: TObject);

var

value: string;

groups: array of groupRecord;

begin

if length(storageFilePath) = 0 then

begin

ShowMessage('Данные не найдены.');

exit;

end;

value := InputBox('Фильтрация по полю "Номер группы"', 'Номер группы:', '');

if length(value) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, Form2.StringGrid1.RowCount - 1);

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

Form2.StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if AnsiLowerCase(Trim(groups[i].number)) = AnsiLowerCase(Trim(value)) then AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.NumberSortAscClick(Sender: TObject);

var

groups: array of groupRecord;

group: groupRecord;

begin

if length(storageFilePath) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, StringGrid1.RowCount - 1);

// Skip first row

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

TArray.Sort<groupRecord>(groups, TDelegatedComparer<groupRecord>.Construct(

function(const Left, Right: groupRecord): integer

begin

Result := TComparer<string>.Default.Compare(left.number, right.number);

end ));

StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.NumberSortDescClick(Sender: TObject);

begin

NumberSortAscClick(NumberSortAsc);

ReverseStringGrid();

end;

procedure TForm2.OpenMenuItemClick(Sender: TObject);

begin

if OpenDialog1.Execute <> true then exit;

UpdateStringGridFromFile(OpenDialog1.FileName);

SetStorageFilePath(OpenDialog1.FileName);

end;

procedure TForm2.PriceFilterMenuItemClick(Sender: TObject);

var

valueNum1, valueNum2: string;

groups: array of groupRecord;

begin

if length(storageFilePath) = 0 then

begin

ShowMessage('Данные не найдены.');

exit;

end;

valueNum1 := InputBox('Фильтрация по полю "Стоимость за занятие"', 'От: ', '');

valueNum2 := InputBox('Фильтрация по полю "Стоимость за занятие"', 'До: ', '');

try

StrToFloat(valueNum1);

StrToFloat(valueNum2);

except

ShowMessage('Введите число!');

exit;

end;

if (length(valueNum1) = 0) or (length(valueNum2) = 0) then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, Form2.StringGrid1.RowCount - 1);

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

Form2.StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if (StrToFloat(groups[i].price) >= StrToFloat(valueNum1)) and (StrToFloat(groups[i].price) <= StrToFloat(valueNum2)) then AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.PriceSortAscClick(Sender: TObject);

var

groups: array of groupRecord;

group: groupRecord;

begin

if length(storageFilePath) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, StringGrid1.RowCount - 1);

// Skip first row

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

TArray.Sort<groupRecord>(groups, TDelegatedComparer<groupRecord>.Construct(

function(const Left, Right: groupRecord): integer

begin

Result := TComparer<real>.Default.Compare(StrToFloat(left.price), StrToFloat(right.price));

end

));

StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.PriceSortDescClick(Sender: TObject);

begin

PriceSortAscClick(PriceSortAsc);

ReverseStringGrid();

end;

procedure TForm2.SaveAsMenuItemClick(Sender: TObject);

begin

if SaveDialog1.Execute <> true then exit;

AssignFile(storageFile, SaveDialog1.FileName);

Rewrite(storageFile);

for i := 1 to StringGrid1.RowCount - 1 do

begin

var group: groupRecord := createGroupFromStringGrid(i);

write(storageFile, group);

end;

SetStorageFilePath(SaveDialog1.FileName);

CloseFile(storageFile);

end;

procedure TForm2.SaveMenuItemClick(Sender: TObject);

begin

if length(storageFilePath) = 0 then

begin

SaveAsMenuItemClick(SaveAsMenuItem);

exit;

end;

AssignFile(storageFile, storageFilePath);

Rewrite(storageFile);

for i := 1 to StringGrid1.RowCount - 1 do

begin

var group: groupRecord := createGroupFromStringGrid(i);

write(storageFile, group);

end;

CloseFile(storageFile);

end;

procedure TForm2.SearchEditChange(Sender: TObject);

var rect: TRect;

begin

StringGrid1.Repaint();

for i := 0 to StringGrid1.ColCount - 1 do

for j := 1 to StringGrid1.RowCount - 1 do

begin

if Pos(AnsiLowerCase(Trim(SearchEdit.Text)), AnsiLowerCase(Trim(StringGrid1.Cells[i, j])))<> 0 then

begin

rect := StringGrid1.CellRect(i, j);

StringGrid1.Canvas.Brush.Color := $C0C0C0;

StringGrid1.Canvas.FillRect(rect);

StringGrid1.Canvas.TextOut(rect.Left + 3, rect.Top + 3, StringGrid1.Cells[i, j]);

end;

end;

end;

procedure TForm2.StringGrid1ContextPopup(Sender: TObject; MousePos: TPoint;

var Handled: Boolean);

var

activeIndex, buttonSelected: integer;

groups: array of groupRecord;

begin

if StringGrid1.Row = 0 then exit;

activeIndex := StringGrid1.Row;

SetLength(groups, StringGrid1.RowCount - 1);

for i := 1 to StringGrid1.RowCount - 1 do

groups[i - 1] := createGroupFromStringGrid(i);

buttonSelected := MessageDlg('Вы точно хотите удалить группу (' + groups[activeIndex - 1].fullName + ')?', mtConfirmation, mbOKCancel, 0);

if buttonSelected <> mrOk then exit;

StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if i + 1 <> activeIndex then AddGroupToStringGrid(groups[i]);

if StringGrid1.RowCount <= 1 then StringGrid1.Options := StringGrid1.Options - [goEditing];

SaveMenuItemClick(SaveMenuItem);

end;

procedure TForm2.StringGrid1DblClick(Sender: TObject);

var group: groupRecord;

begin

if StringGrid1.Row = 0 then exit;

group := createGroupFromStringGrid(StringGrid1.Row);

with Form3 do

begin

Caption := 'Редактировать группу';

NumberComboBox.ItemIndex:=NumberComboBox.Items.IndexOf(group.number);

LevelComboBox.Text := group.level;

FullnameEdit.Text := group.fullName;

PriceEdit.Text := group.price;

DateTimePicker1.Date := StrToDate(group.day);

DateTimePicker2.Time := StrToTime(group.time);

CountEdit.Text := group.studentsCount;

ShowModal;

group.number := NumberComboBox.Text;

group.level := LevelComboBox.Text;

group.fullName := FullnameEdit.Text;

group.price := PriceEdit.Text;

group.day := DateToStr(DateTimePicker1.Date);

group.time := TimeToStr(DateTimePicker2.Time);

group.studentsCount := CountEdit.Text;

end;

StringGrid1.Cells[0, StringGrid1.Row] := group.number;

StringGrid1.Cells[1, StringGrid1.Row] := group.level;

StringGrid1.Cells[2, StringGrid1.Row] := group.fullName;

StringGrid1.Cells[3, StringGrid1.Row] := group.price;

StringGrid1.Cells[4, StringGrid1.Row] := group.day;

StringGrid1.Cells[5, StringGrid1.Row] := group.time;

StringGrid1.Cells[6, StringGrid1.Row] := group.studentsCount;

end;

procedure TForm2.StudentsCountFilterClick(Sender: TObject);

var

valueNum1, valueNum2: string;

groups: array of groupRecord;

begin

if length(storageFilePath) = 0 then

begin

ShowMessage('Данные не найдены.');

exit;

end;

valueNum1 := InputBox('Фильтрация по полю "Количество учащихся"', 'От: ', '');

valueNum2 := InputBox('Фильтрация по полю "Количество учащихся"', 'До: ', '');

try

StrToFloat(valueNum1);

StrToFloat(valueNum2);

except

ShowMessage('Введите число!');

exit;

end;

if (length(valueNum1) = 0) or (length(valueNum2) = 0) then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, Form2.StringGrid1.RowCount - 1);

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

Form2.StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if (StrToFloat(groups[i].studentsCount) >= StrToFloat(valueNum1)) and (StrToFloat(groups[i].studentsCount) <= StrToFloat(valueNum2)) then

AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.StudentsCountSortAscClick(Sender: TObject);

var

groups: array of groupRecord;

group: groupRecord;

begin

if length(storageFilePath) = 0 then exit;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, StringGrid1.RowCount - 1);

// Skip first row

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

TArray.Sort<groupRecord>(groups, TDelegatedComparer<groupRecord>.Construct(

function(const Left, Right: groupRecord): integer

begin

Result:=TComparer<integer>.Default.Compare(StrToInt(left.studentsCount), StrToInt(right.studentsCount));

end

));

StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

AddGroupToStringGrid(groups[i]);

end;

procedure TForm2.StudentsCountSortDescClick(Sender: TObject);

begin

StudentsCountSortAscClick(StudentsCountSortAsc);

ReverseStringGrid();

end;

end.

Текст модуля «Unit3»:

unit Unit3;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ComCtrls, Vcl.Grids, Unit2;

type

TForm3 = class(TForm)

Label1: TLabel;

Label2: TLabel;

FullnameEdit: TEdit;

Label3: TLabel;

PriceEdit: TEdit;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

CountEdit: TEdit;

Label7: TLabel;

AddGroupButton: TButton;

DateTimePicker1: TDateTimePicker;

DateTimePicker2: TDateTimePicker;

NumberComboBox: TComboBox;

LevelComboBox: TComboBox;

procedure AddGroupButtonClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure PriceEditKeyPress(Sender: TObject; var Key: Char);

procedure FullnameEditKeyPress(Sender: TObject; var Key: Char);

private

{ Private declarations }

public

{ Public declarations }

end;

procedure AddGroupToStringGrid(group: groupRecord);

var

Form3: TForm3;

implementation

{$R \*.dfm}

function IsRussianChars(const S: String): Boolean;

var

i: Integer;

begin

Result:= True;

for i:= 1 to Length(S) do

if (Ord(S[i]) >= $C0)

or (Ord(S[i]) = $A8)

or (Ord(S[i]) = $B8)

then Exit;

Result:= False;

end;

procedure AddGroupToStringGrid(group: groupRecord);

begin

Form2.StringGrid1.RowCount := Form2.StringGrid1.RowCount + 1;

Form2.StringGrid1.FixedRows := 1;

const rowIndex = Form2.StringGrid1.RowCount - 1;

Form2.StringGrid1.Cells[0, rowIndex] := group.number;

Form2.StringGrid1.Cells[1, rowIndex] := group.level;

Form2.StringGrid1.Cells[2, rowIndex] := group.fullName;

Form2.StringGrid1.Cells[3, rowIndex] := group.price;

Form2.StringGrid1.Cells[4, rowIndex] := group.day;

Form2.StringGrid1.Cells[5, rowIndex] := group.time;

Form2.StringGrid1.Cells[6, rowIndex] := group.studentsCount;

end;

procedure TForm3.AddGroupButtonClick(Sender: TObject);

var

group: groupRecord;

i: integer;

begin

for i := 0 to Form3.ComponentCount - 1 do

if Form3.Components[i] is TEdit then

if TEdit(Form3.Components[i]).Text = '' then

begin

ShowMessage('Заполните все поля!');

exit;

end;

group.number := NumberComboBox.Text;

group.level := LevelComboBox.Text;

group.fullName := FullnameEdit.Text;

group.price := PriceEdit.Text;

group.day := DateToStr(DateTimePicker1.Date);

group.time := TimeToStr(DateTimePicker2.Time);

group.studentsCount := CountEdit.Text;

if Pos('добавить', AnsiLowerCase(Form3.Caption)) <> 0 then AddGroupToStringGrid(group);

Form3.Close;

Form2.SaveMenuItemClick(Form2.SaveMenuItem);

end;

procedure TForm3.FormCreate(Sender: TObject);

begin

Form3.Color :=$C0DCC0;

end;

procedure TForm3.FullnameEditKeyPress(Sender: TObject; var Key: Char);

begin

if not ((Key in [' ', '-', #8]) or (IsRussianChars(Key))) then Key := #0;

end;

procedure TForm3.PriceEditKeyPress(Sender: TObject; var Key: Char);

begin

if not (Key in ['0'..'9', ',', #8]) then Key := #0;

end;

end.

Текст модуля «Unit4»:

unit Unit4;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, DateUtils, System.Generics.Collections,

System.Generics.Defaults;

type

TForm4 = class(TForm)

ListBox1: TListBox;

ApplyFilterButton: TButton;

procedure ApplyFilterButtonClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form4: TForm4;

implementation

uses Unit2, Unit3;

{$R \*.dfm}

procedure TForm4.ApplyFilterButtonClick(Sender: TObject);

var

i, activeIndex: integer;

groups: array of groupRecord;

group: groupRecord;

begin

if ListBox1.ItemIndex = -1 then

begin

ShowMessage('Выберите день недели.');

exit;

end;

if length(storageFilePath) = 0 then

begin

ShowMessage('Данные не найдены.');

exit;

end;

UpdateStringGridFromFile(storageFilePath);

SetLength(groups, Form2.StringGrid1.RowCount - 1);

activeIndex := ListBox1.ItemIndex + 1;

// Skip first row

for i := 1 to Length(groups) do

groups[i - 1] := createGroupFromStringGrid(i);

Form2.StringGrid1.RowCount := 1;

for i := 0 to Length(groups) - 1 do

if DayOfTheWeek(StrToDateTime(groups[i].day)) = activeIndex then

AddGroupToStringGrid(groups[i]);

end;

procedure TForm4.FormCreate(Sender: TObject);

begin

Form4.Color :=$C0DCC0;

end;

end.

Текст модуля «Unit5»:

unit Unit5;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type

TForm5 = class(TForm)

ListBox1: TListBox;

ApplyFilterButton: TButton;

Label1: TLabel;

procedure FormActivate(Sender: TObject);

procedure ApplyFilterButtonClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form5: TForm5;

implementation

uses Unit2, Unit3;

{$R \*.dfm}

procedure TForm5.ApplyFilterButtonClick(Sender: TObject);

var

i: integer;

price: real;

group: groupRecord;

begin

if ListBox1.ItemIndex = -1 then

begin

ShowMessage('Выберите уровень.');

exit;

end;

if Form2.StringGrid1.RowCount <= 1 then

begin

ShowMessage('Данные не найдены.');

exit;

end;

// Skip first row

for i := 1 to Form2.StringGrid1.RowCount - 1 do

begin

group := createGroupFromStringGrid(i);

if group.level <> ListBox1.Items[ListBox1.ItemIndex] then continue;

price := StrToFloat(group.price);

Form2.StringGrid1.Cells[3, i] := FloatToStr(round(price + (price \* 10 / 100)));

end;

end;

procedure TForm5.FormActivate(Sender: TObject);

var i: integer;

begin

ListBox1.Items.Clear;

for i := 0 to Length(levels) - 1 do

ListBox1.Items.Add(levels[i]);

end;

procedure TForm5.FormCreate(Sender: TObject);

begin

Form5.Color :=$C0DCC0;

end;

end.

Текст модуля «Unit6»:

unit Unit6;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ComCtrls;

type

TDateRangeInputForm = class(TForm)

Label1: TLabel;

DateTimePicker1: TDateTimePicker;

Label2: TLabel;

DateTimePicker2: TDateTimePicker;

Button1: TButton;

procedure Button1Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

DateRangeInputForm: TDateRangeInputForm;

implementation

{$R \*.dfm}

procedure TDateRangeInputForm.Button1Click(Sender: TObject);

begin

DateRangeInputForm.Close;

end;

procedure TDateRangeInputForm.FormCreate(Sender: TObject);

begin

DateRangeInputForm.Color:= $C0DCC0;

end;

end.