Частное учреждение образования

«Колледж бизнеса и права»

«СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ДЛЯ МОНИТОРИНГА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ В РАМКАХ УЧЕБНОГО ПРЕДМЕТЫ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Основы алгоритмизации и программирования»

КП Т.295010.401

Руководитель проекта (Дж. А. Миронова)

Учащийся (В. И. Колосовский)

2024

Изм.

Лист

№ докум.

Подпись

Датаа

Лист

3

КП Т.295010.401 ПЗ

Разраб.

Колосовский В.И.

Провер.

Миронова Дж.А.

Реценз.

Н. Контр.

Утверд.

СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ДЛЯ МОНИТОРИНГА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ В РАМКАХ УЧЕБНОГО ПРЕДМЕТА «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»»

Лит.

Листов

КБиП

50

У

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc170271216)

[1 Описание задачи 5](#_Toc170271217)

[1.1 Анализ предметной области 5](#_Toc170271218)

[1.2 Постановка задачи 6](#_Toc170271219)

[2 Проектирование системы 7](#_Toc170271220)

[2.1 Требования к приложению 7](#_Toc170271221)

[2.2 Проектирование модели 7](#_Toc170271222)

[2.3 Концептуальный прототип 8](#_Toc170271223)

[3 Описание реализации программного средства 11](#_Toc170271224)

[3.1 Инструменты разработки и применяемые технологии 11](#_Toc170271225)

[3.2 Функции: логическая и физическая организация 11](#_Toc170271226)

[3.3 Входные и выходные данные 15](#_Toc170271227)

[3.4 Функциональное тестирование 15](#_Toc170271228)

[3.5 Описание справочной системы 17](#_Toc170271229)

[4 Применение 18](#_Toc170271230)

[4.1 Назначение программного средства 18](#_Toc170271231)

[4.2 Условия применения 18](#_Toc170271232)

[Заключение 19](#_Toc170271233)

[Список использованных источников 20](#_Toc170271234)

[Приложение А 21](#_Toc170271235)

[Приложение Б 48](#_Toc170271236)

Введение

В наш век высоких технологий, когда информационные системы становятся незаменимыми помощниками, особенно актуальным становится использование программных средств в образовательном процессе. Эти инструменты значительно облегчают и ускоряют обучение, делая его более интерактивным и результативным.

Целью разработки курсового проекта на тему «Создание программного средства для мониторинга выполнения лабораторных работ» является реализация приложения, которое будет облегчать управление и отслеживание информации о лабораторных работах, предоставлять возможности для поиска и сортировки данных, а также формировать и экспортировать отчеты, обладая при этом простым и интуитивно понятным интерфейсом.

Программное средство для мониторинга выполнения лабораторных работ представляет собой не просто инструмент, но и верного союзника для студентов и преподавателей. Оно предоставляет возможность не только следить за прогрессом выполнения работ, но и выполнять множество других функций, что делает процесс обучения максимально прозрачным и помогает студентам эффективно планировать свое учебное время.

Актуальность данной темы подчеркивается не только необходимостью автоматизации процесса управления лабораторными работами, но и стремлением к оптимизации времени, которое является одним из самых ценных ресурсов в жизни современного человека. Автоматизация управления лабораторными работами позволяет не только сэкономить время, но и значительно усилить контроль над процессом обучения, повышая его качество и эффективность.

В первом разделе «Описание задачи» рассматривается актуальность поставленной задачи и анализ предметной области.

Второй раздел «Проектирование системы» посвящен требованиям программного средства, проектированию модели, концептуальному прототипу

Третий раздел «Описание реализации программного средства» описывает функциональное назначение, справочную систему, структуру входных и выходных данных приложения

Четвертый раздел «Применение» посвящен условиям применения приложения и его назначения.

В заключении проводится анализ результатов работы и сделаны выводы о достигнутых результатах и перспективах дальнейшего развития проекта.

1 Описание задачи

1.1 Анализ предметной области

Целью данного проекта является разработка программного средства для мониторинга и управления процессом выполнения лабораторных работ студентами в рамках курса «Основы алгоритмизации и программирования». Программное средство должно содержать удобный функционал для просмотра, добавления, редактирования и удаления информации о сдаче лабораторных работ.

Предметная область включает в себя комплекс учебных и методических процессов, направленных на обучение студентов основам создания алгоритмов и программирования. В рамках учебного предмета «Основы алгоритмизации и программирования», студенты выполняют лабораторные работы, которые являются ключевым элементом практического применения теоретических знаний и требуют четкой организации процесса сдачи, проверки и оценки. Традиционно, этот процесс включает в себя ручное управление списками студентов, их работами и результатами, что может привести к ошибкам и затруднениям в отслеживании прогресса каждого студента.

Процесс сдачи лабораторных работ в рамках курса «Основы алгоритмизации и программирования» представляет собой последовательность шагов, начиная с подготовки и распределения заданий и заканчивая мониторингом и анализом результатов. Преподаватели предоставляют лабораторные работы студентам, устанавливая четкие сроки для выполнения. Студенты, в свою очередь, разрабатывают алгоритмы, пишут код и тестируют его, стремясь соответствовать заданным задачам. После этого они сдают выполненные работы на проверку, где преподаватели оценивают качество и корректность выполнения. Важной частью процесса является обратная связь и оценки, которые преподаватели предоставляют студентам, помогая им улучшить навыки и знания. Наконец, преподаватели отслеживают прогресс студентов и анализируют собранные данные, что позволяет улучшить учебный процесс и адаптировать его под нужды студентов.

Компьютерная обработка информации значительно упрощает управление данными о сдаче лабораторных работ, повышает точность и надежность хранения информации, а также обеспечивает быстрый доступ к данным и их анализ. В отличие от бумажных носителей, программа позволяет мгновенно обновлять и редактировать информацию, а также предоставляет возможность множественного поиска и сортировки данных по различным параметрам.

Потенциальными пользователями программного средства являются преподаватели, которые участвуют в процессе обучения по курсу «Основы алгоритмизации и программирования». Программное средство позволит им эффективно управлять процессом сдачи и проверки лабораторных работ, экономя время и ресурсы.

1.2 Постановка задачи

Задачей проекта является разработка программного средства для мониторинга и управления процессом выполнения лабораторных работ студентами в рамках курса «Основы алгоритмизации и программирования». Это приложение будет обеспечивать авторизацию пользователей, предоставлять инструменты для просмотра, добавления, редактирования и удаления информации о сдаче лабораторных работ, а также разработку механизма отслеживания сроков сдачи работ.

Программное средство будет включать в себя функционал для множественного поиска и сортировки информации по различным параметрам, что позволит преподавателям эффективно ориентироваться в большом объеме данных. Кроме того, будет реализована возможность формирования отчетов о сдаче лабораторных работ за выбранный промежуток времени и их экспорт.

На рисунке 1.1 представлен аналог разрабатываемого приложения «Triptonkosti»

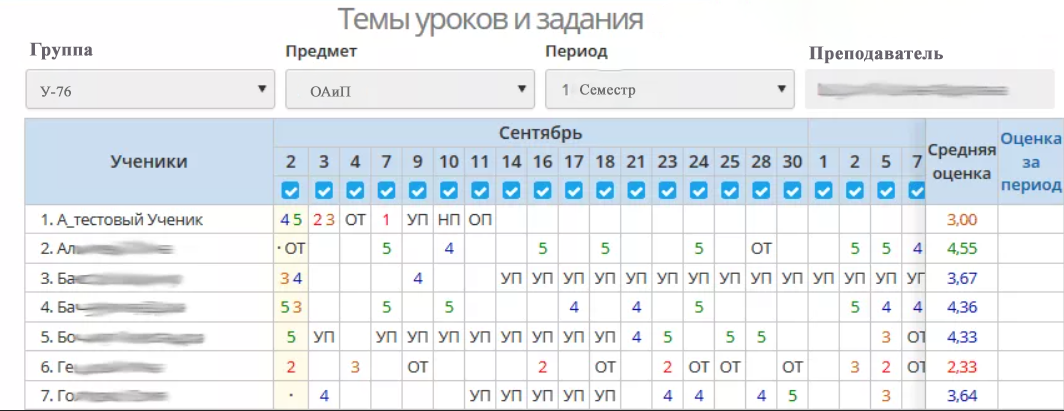


Рисунок 1.1 – Внешний вид приложения «Triptonkosti»

Достоинства аналога: данные упорядочены и легко читаемы, использование цветовых кодов для различных типов оценок, предоставление детальной информации о успеваемости каждого ученика

Недостатки аналога: отсутствие возможности формирования и экспорта отчётов сдачи лабораторных работ за выбранный промежуток времени

В разрабатываемом программном средстве будут устранены недостатки аналога приложения.

2 Проектирование системы

2.1 Требования к приложению

Для обеспечения безопасности данных, приложение включает систему авторизации с использованием логина и пароля. Учетные данные пользователя хранятся в файле с расширением .txt, что обеспечивает первичный уровень защиты от несанкционированного доступа.

Приложение состоит из двух основных форм: формы авторизации и формы мониторинга лабораторных работ. Форма авторизации представляет собой простой и интуитивно понятный интерфейс, где пользователь вводит свои учетные данные. Форма мониторинга лабораторных работ дает обзор текущего статуса выполнения заданий и предоставляет функционал для управления процессом обучения.

Интерфейс приложения разработан с учетом удобства использования и читаемости. Элементы управления имеют стандартный размер шрифта 14 и выполнены в едином стиле с использованием шрифта Bahnschrift. Расположение элементов продумано таким образом, чтобы обеспечить логичный и эффективный рабочий процесс.

Минимальными требованиями к аппаратному и программному обеспечению, необходимыми для корректной работы программы являются:

* операционная система Windows 10/11;
* процессор AMD Athlon 1800 МГц и выше;
* объем оперативной памяти 1024 МБ и более.
* клавиатура
* компьютерная мышь

Наиболее удобной операционной системой для проведения испытаний является Windows 10, так как она ориентирована на максимальное использование всех возможностей персонального компьютера и обеспечение комфортных условий работы.

2.2 Проектирование модели

Диаграмма работы алгоритма отслеживания срока сдачи лабораторных работ — это визуальное представление последовательности операций, выполняемых алгоритмом отслеживания сроков сдачи лабораторных работ [2]. Эта диаграмма необходима для понимания логики работы алгоритма и его взаимодействия с пользователем и данными. Она помогает разработчикам и аналитикам увидеть, как алгоритм обрабатывает информацию о сроках и статусах сдачи работ. Основные компоненты диаграммы — это блоки операций, условные переходы, циклы и точки входа/выхода данных.

Диаграмма работы алгоритмов представлена в графической части на листе 1.

Диаграмма классов — это структурное представление классов, их атрибутов, методов и взаимосвязей между ними в контексте программного средства [3]. Она необходима для проектирования архитектуры программного обеспечения, определения структуры данных и способов их обработки. Также диаграмма классов служит основой для написания кода и документации. Диаграмма включает в себя классы с указанием их атрибутов и методов, а также связи, такие как наследование, ассоциации, агрегация и композиция.

Диаграмма работы алгоритмов представлена в графической части на листе 2.

2.3 Концептуальный прототип

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления.

Основной интерфейс программного средства будут представлять окна. Все окна будут содержать стандартные пользовательские элементы управления.

В программе предполагается создание навигационных кнопок.

На рисунке 2.1 представлен прототип окна авторизации

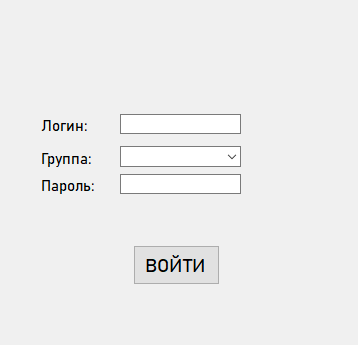


Рисунок 2.1 – Прототип окна авторизации

Кнопка «ВОЙТИ» будет перенаправлять преподавателя в главное меню, но только при заполнении всех обязательных полей валидными данными.

Прототип окна, служащего главным меню, представлен на рисунке 2.2.

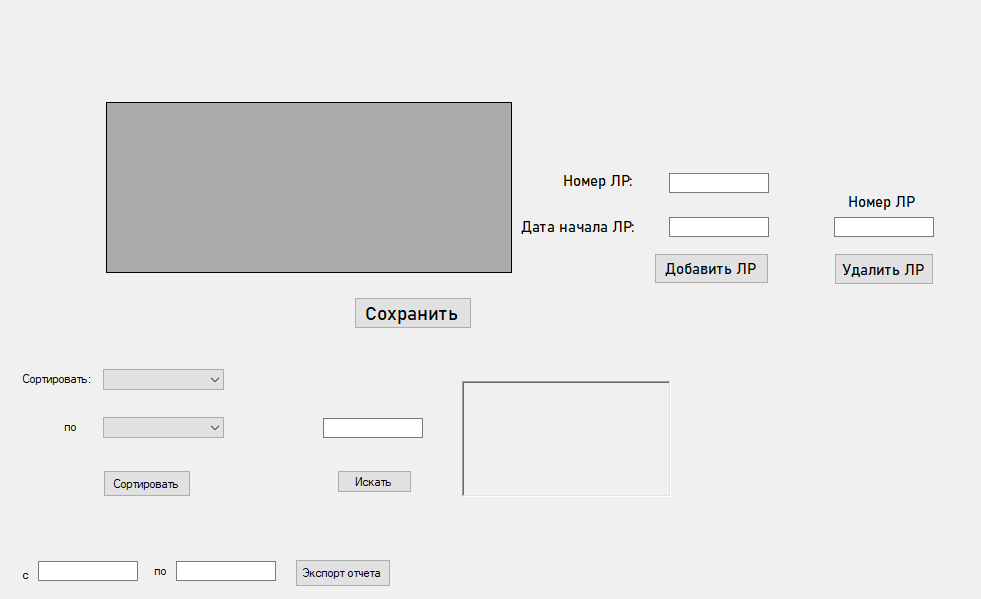


Рисунок 2.2 – Прототип окна главного меню

Кнопка «Добавить ЛР» будет добавлять информацию о лабораторной работе в таблицу, но только при заполнении всех обязательных полей «Номер ЛР», «Дата начала ЛР» валидными данными.



Рисунок 2.3 – Кнопка «Добавить ЛР»

Кнопка «Удалить ЛР» будет удалять информацию о лабораторной работе из таблицы.



Рисунок 2.4 – Кнопка «Удалить ЛР»

Кнопка «Сохранить» будет сохранять всю информацию в таблице.



Рисунок 2.5 – Кнопка «Сохранить»

Кнопка «Сортировать» будет сортировать информацию в таблице по различным параметрам.



Рисунок 2.6 – Кнопка «Сортировать»

Кнопка «Искать» будет искать информацию из таблицы и выводить её в текстовое поле.



Рисунок 2.7 – Кнопка «Искать»

Кнопка «Экспорт отчета» будет экспортировать отчет пользователю в любое удобное ему место.



Рисунок 2.8 – Кнопка «Экспорт отчета»

3 Описание реализации программного средства

3.1 Инструменты разработки и применяемые технологии

Для разработки данного программного средства был выбран язык программирования C++, среда разработки Microsoft Visual Studio, включая поддержку Windows Forms и фреймворка .NET.

C++ - это мощный язык программирования, который поддерживает различные парадигмы, включая процедурное, объектно-ориентированное и обобщенное программирование [4]. Он широко используется для разработки сложных системных приложений, игр и встроенных систем.

Microsoft Visual Studio - это интегрированная среда разработки (IDE), которая предоставляет разработчикам набор инструментов для создания приложений для Windows, Android и iOS, а также веб-сайтов, веб-приложений и облачных служб [5].

Windows Forms - это технология Microsoft, которая используется для создания пользовательских интерфейсов в приложениях Windows [6]. Она предоставляет богатый набор графических элементов управления, которые можно использовать для создания пользовательских форм, диалоговых окон и полнофункциональных приложений с графическим интерфейсом.

Примеры приложений с использованием Windows Forms:

1. Microsoft Visual Studio: Сама среда разработки Visual Studio является примером приложения, созданного с использованием Windows Forms.
2. Paint.NET: Это бесплатное и открытое графическое приложение для Windows, созданное с использованием Windows Forms.

Windows Forms используется в проектируемом программном обеспечении для создания пользовательского интерфейса. Это позволяет разработчикам создавать интуитивно понятные и привлекательные интерфейсы, которые упрощают взаимодействие пользователя с приложением.

Windows Forms был выбран из-за его богатого набора элементов управления и простоты использования. Он позволяет быстро создавать сложные пользовательские интерфейсы с минимальными усилиями. Это делает Windows Forms идеальным выбором для создания настольных приложений для Windows.

3.2 Функции: логическая и физическая организация

Функция button1\_Click обрабатывает событие нажатия кнопки для добавления новой лабораторной работы. Она проверяет корректность введенных данных, создает новый столбец в таблице для новой лабораторной работы и сохраняет обновленную таблицу в файл.

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

try {

if (textBox1->Text != "") {

String^ labNumber = textBox1->Text;

System::Text::RegularExpressions::Regex^ integerRegex = gcnew System::Text::RegularExpressions::Regex("^[0-9]+$");

if (!integerRegex->IsMatch(labNumber)) {

MessageBox::Show("В поле 'Номер ЛР' можно вводить только целые числа");

textBox1->Clear();

return;

}

// Проверка на существование номера лабораторной работы

for (int i = 0; i < dataGridView1->Columns->Count; ++i) {

if (dataGridView1->Columns[i]->Name->Contains("ЛР" + labNumber)) {

MessageBox::Show("Такой номер лабораторной работы уже существует");

textBox1->Clear();

return;

; }

}

if (textBox2->Text != "") {

String^ startDate = textBox2->Text;

// Проверка формата даты

System::Text::RegularExpressions::Regex^ dateRegex = gcnew System::Text::RegularExpressions::Regex("^\\d{2}\\.\\d{2}$");

if (!dateRegex->IsMatch(startDate)) {

MessageBox::Show("Дата начала ЛР введена в неправильном формате. Формат должен быть 01.01 (число.месяц)");

textBox2->Clear();

return;

}

// Расчет конечной даты

array<String^>^ dateParts = startDate->Split('.');

int day = Int32::Parse(dateParts[0]);

int month = Int32::Parse(dateParts[1]);

DateTime startDateTime = DateTime(DateTime::Now.Year, month, day);

DateTime endDateTime = startDateTime.AddDays(14);

String^ endDate = endDateTime.ToString("dd.MM");

String^ columnName = "ЛР" + labNumber + " " + startDate + " (" + endDate + ")";

dataGridView1->Columns->Add(columnName, columnName);

// Заполните новый столбец пустыми значениями

for each (DataGridViewRow ^ row in dataGridView1->Rows) {

row->Cells[columnName]->Value = "";

}

CheckDatesAndColorCells(dataGridView1);

StreamReader^ group = File::OpenText("current.txt");

String^ name = group->ReadToEnd();

group->Close();

name = name->Trim();

String^ tex = Convert::ToString(name) + ".csv";

SaveDataToCSV(dataGridView1, tex);

}

else {

MessageBox::Show("Введите дату начала ЛР");

}

}

else {

MessageBox::Show("Введите номер лабораторной работы");

}

}

catch (Exception^ ex) {

MessageBox::Show("Введена некорректная дата");

textBox2->Clear();

}

}

Функция button3\_Click обрабатывает событие нажатия кнопки для сохранения текущего состояния таблицы. Она сохраняет данные из таблицы в файл.

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

StreamReader^ group = File::OpenText("current.txt");

String^ name = group->ReadToEnd();

group->Close();

name = name->Trim();

String^ tex = Convert::ToString(name) + ".csv";

SaveDataToCSV(dataGridView1,tex);

CheckDatesAndColorCells(dataGridView1);

}

Функция button4\_Click обрабатывает событие нажатия кнопки для удаления лабораторной работы. Она проверяет корректность введенного номера лабораторной работы и удаляет соответствующий столбец из таблицы.

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

if (textBox4->Text != "") {

String^ labNumber = textBox4->Text;

System::Text::RegularExpressions::Regex^ integerRegex = gcnew System::Text::RegularExpressions::Regex("^[0-9]+$");

if (!integerRegex->IsMatch(labNumber)) {

MessageBox::Show("В поле 'Номер ЛР' можно вводить только целые числа");

textBox4->Clear();

return;

}

if (textBox4->Text != "" && (Convert::ToInt32(textBox4->Text) > 0)) {

String^ labNumber = textBox4->Text;

int columnIndex = -1;

// Проверка на существование номера лабораторной работы

for (int i = 0; i < dataGridView1->Columns->Count; ++i) {

if (dataGridView1->Columns[i]->Name->Contains("ЛР" + labNumber)) {

columnIndex = i;

break;

}

}

if (columnIndex != -1) {

dataGridView1->Columns->RemoveAt(columnIndex);

}

else {

MessageBox::Show("Лабораторная работа с номером " + labNumber + " не найдена.");

}

}

else {

MessageBox::Show("Введите корректный номер лабораторной работы для удаления");

this->textBox4->Clear();

}

}

else {

MessageBox::Show("Введите номер лабораторной работы");

}

}

Полный текст программных модулей представлен в приложении А.

3.3 Входные и выходные данные

Входные данные:

Логин – строка, представляющая фамилию преподавателя, Группа – строка, представляющая номер группы, Пароль – строка, представляющая пароль от учетной записи преподавателя, Номер ЛР – строка, представляющая номер лабораторной работы из таблицы, Дата начала ЛР – строка, представляющая дату начала лабораторной работы, Параметр сортировки – строка, представляющая параметр сортировки таблицы, Принцип сортировки – строка, представляющая принцип сортировки таблицы, Запрос поиска – строка, представляющая критерий поиска, Начальная дата формирования отчета – строка, представляющая начальную дату для формирования отчета, Конечная дата формирования отчета – строка, представляющая конечная дату для формирования отчета

Выходные данные:

Столбец таблицы – столбец таблицы dataGridView, содержащий информацию о лабораторной работе, Результат поиска – строка, представляющая результат поиска по запросу пользователя, Файл отчёта – файл, содержащий номера лабораторных работ, фамилии учащихся, которые не сдали лабораторную работу, фамилии учащихся, которые сдали лабораторную работу, оценки учащихся за лабораторную работу

3.4 Функциональное тестирование

Функциональное тестирование – это вид тестирования, который проверяет соответствие функциональности продукта тому, как он был задуман. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тест-кейсы для основных функций представлены в таблице 3.1.

Таблица 3.1 – Функциональное тестирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №№ | Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 11 | Авторизация | 1. Заполняем поля данными:   * Логин: «Булькова» * Пароль: «123456»   2. Выбираем группу «Т-444» из раскрывающегося меню  3. Нажать на кнопку «ВОЙТИ» | Вывод предупреждения и открытие окна главного меню | Вывод предупреждения и открытие окна главного меню.  Результат представлен на рисунке Б.1 в приложении Б |
| 22 | Авторизация | 1. Заполняем поля данными:   * Логин: не заполняем * Пароль: «123456»   2. Выбираем группу «Т-444» из раскрывающегося меню  3. Нажать на кнопку «ВОЙТИ» | Вывод предупреждения | Вывод предупреждения.  Результат представлен на рисунке Б.2 в приложении Б |
| 23 | Добавление лабораторной работы | 1. Заполняем поля данными:   * Номер ЛР: «8» * Дата начала ЛР: «12.06»   2. Нажать на кнопку «Добавить ЛР» | Добавление лабораторной работы в таблицу | Добавление лабораторной работы в таблицу. Результат представлен на рисунке Б.3 в приложении Б |
| 24 | Добавление лабораторной работы | 1. Заполняем поля данными:   * Номер ЛР: «9» * Дата начала ЛР: «123»   2. Нажать на кнопку «Добавить ЛР» | Вывод предупреждения | Вывод предупреждения. Результат представлен на рисунке Б.4 в приложении Б |
| 25 | Сортировка | 1. Выбираем параметр сортировки «лабораторные работы» из раскрывающегося меню  2. Выбираем принцип сортировки «убыванию» из раскрывающегося меню  3. Нажать на кнопку «Сортировать» | Сортировка столбцов таблицы | Сортировка столбцов таблицы.  Результат представлен на рисунке Б.5 в приложении Б |
| 66 | Формирование отчетов | 1. Заполняем поля данными:   * с: «13.03» * по: «17.05»   2. Нажать на кнопку «Экспорт отчета» | Сохранение файла отчета, вывод предупреждения | Сохранение файла отчета, вывод предупреждения.  Результат представлен на рисунке Б.6 и Б.7 в приложении Б |
| 7 | Формирование отчетов | 1. Заполняем поля данными:   * с: «123» * по: «17.05»   2. Нажать на кнопку «Экспорт отчета» | Вывод предупреждения | Вывод предупреждения.  Результат представлен на рисунке Б.8 в приложении Б |
| 8 | Поиск информации | 1. Заполняем поле поиска фамилией из таблицы  2. Нажать на кнопку «Искать» | Вывод информации в текстовое поле | Вывод информации в текстовое поле.  Результат представлен на рисунке Б.9 в приложении Б |
| 9 | Поиск информации | 1. Заполняем поле поиска значением «123»  2. Нажать на кнопку «Искать» | Вывод информации в текстовое поле | Вывод информации в текстовое поле.  Результат представлен на рисунке Б.10 в приложении Б |

3.5 Описание справочной системы

Средством разработки для создания справочной системы является программное средство Dr.Explain, которое помогает создавать файлы справки, документацию и руководства пользователя [7].

Справочная система состоит из начальной страницы, оглавления, 9-ти разделов и 3-х подразделов.

Раздел 1 «Запуск приложения» Предоставляет описание процесса запуска приложения в системе.

Подраздел 1.1 «Шаги по установке Python» является инструкцией по установке Python, для работы скрипта программного средства.

Раздел 2 «Авторизация» Является подробным обзором страницы входа.

Раздел 3 «Регистрация» Предоставляет инструкцию по регистрации.

Подраздел 3.1 «Заполнение профиля» Предоставляет подробную инструкцию по начальному заполнению профиля и личной информации. Подраздел 3.2 «Начальная настройка» Предоставляет инструкцию по ранней настройке внешнего вида приложения.

Раздел 4 «Главная страница» Предоставляет информацию об элементах на главной странице программного средства.

Раздел 5 «Книга» Предоставляет инструкцию по использованию функций книги.

Раздел 6 «Задания» Предоставляет подробную информацию о странице задач и о самих задачах.

Раздел 7 «Профиль» Предоставляет пользователю информацию о его личных данных и профиле, а также возможности редактирования этих данных.

Раздел 8 «Статистика» Обозревает функции страницы статистики.

Раздел 9 «Страница настроек» Явно показывает функционал и советы по настройке.

Справочная система вызывается c помощью нажатия клавиши F1 на клавиатуре.

4 Применение

4.1 Назначение программного средства

Программное средство «163umneck» разработано для мониторинга выполнения лабораторных работ в рамках учебного предмета «Основы алгоритмизации и программирования». Пользователи могут использовать это программное средство для просмотра, добавления, изменения, удаления, сортировки, поиска информации о лабораторных работах и формирования отчетов сдачи лабораторных работ за выбранный промежуток времени.

Ограничения на область применения программного средства включают в себя то, что оно предназначено только для мониторинга сдачи лабораторных работ и не предназначено для управления другими аспектами учебного процесса.

4.2 Условия применения

Для функционирования программного средства «163umneck» требуется операционная система Windows (рекомендуется использовать версии 7, 8, 10). Процесс запуска приложения заключается в перемещения содержимого папки «kurs» в выбранную директорию на компьютере и запуске файла «163umneck.exe».

Заключение

Целью курсового проектирования было создание программного средства, программное средство для мониторинга выполнения лабораторных работ в рамках учебного предмета «Основы алгоритмизации и программирования».

Для достижения поставленной цели были решены следующие задачи: исследована предметная область, составлен концептуальный прототип, проанализированы существующие аналоги, проанализированы требования к аппаратным и операционным ресурсам, проведено функциональное тестирование.

Программное средство было реализовано в среде разработки Visual Studio на языке программирования C++, используя программирования приложений Windows Forms.

Программное средство «163umneck» предназначено для просмотра, добавления, изменения, удаления, сортировки, поиска информации о лабораторных работах и формирования отчетов сдачи лабораторных работ за выбранный промежуток времени.

В процессе разработки возникли трудности с реализацией интуитивно понятного и удобного пользовательского интерфейса, но они были преодолены с помощью удобного интерфейса программирования приложений Windows Forms.

Преимуществами данного программного средства являются интуитивно понятный и удобный интерфейс приложения, что облегчает взаимодействие с приложением. Это программное средство предлагает широкий спектр функций, включая просмотр, добавление, изменение, удаление, сортировку и поиск информации о лабораторных работах, а также формирование отчетов.

Недостатками является то, что программное средство предназначено только для мониторинга сдачи лабораторных работ и не может быть использовано для управления другими аспектами учебного процесса, а также возможность работы программного средства только на операционной системе Windows и невозможность работы на других операционных системах.

Улучшение данного программного средства заключается в расширении функциональности. Возможно добавить новые функции, такие как автоматическое напоминание о сроках сдачи работ, интеграцию с электронной почтой для отправки уведомлений и отчетов. Также возможно разработать мобильное приложение, которое будет синхронизироваться с программным средством, чтобы пользователи могли мониторить сдачу лабораторных работ на своих мобильных устройствах. Этого можно достичь используя фреймворки для разработки мобильных приложений, такие как React Native или Flutter.

Список использованных источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.
2. Блок-схема [Электронный ресурс]. – Wikipedia, 2024. – Режим доступа : https://ru.wikipedia.org/wiki/Блок-схема. – Дата доступа : 15.04.2024.
3. Диаграмма классов [Электронный ресурс]. – Wikipedia, 2024. – Режим доступа : https://ru.wikipedia.org/wiki/Диаграмма\_классов. – Дата доступа : 17.04.2024.
4. C++ [Электронный ресурс]. – Wikipedia, 2024. - Режим доступа : https://ru.wikipedia.org/wiki/C++. – Дата доступа : 11.05.2024.
5. Visual Studio 2022 [Электронный ресурс]. – Microsoft, 2024. – Режим доступа : https://visualstudio.microsoft.com/ru/vs. – Дата доступа : 13.05.2024.
6. Windows Forms [Электронный ресурс]. – Wikipedia, 2024. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/. Дата доступа : 15.05.2024.
7. Dr.Explain [Электронный ресурс]. – drexplain.ru, 2024. – Режим доступа: https://www.drexplain.ru/. – Дата доступа : 25.05.2024.

Приложение А

**(обязательное)**

**Текст программных модулей**

«auth.cpp»

#include "auth.h"

#include <Windows.h>

using namespace kurs;

[STAThreadAttribute]

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Application::Run(gcnew auth);

return 0;

}

«labs.cpp»

#include "labs.h"

«auth.h»

#pragma once

#include "labs.h"

namespace kurs {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::IO;

/// <summary>

/// Сводка для auth

/// </summary>

public ref class auth : public System::Windows::Forms::Form

{

public:

auth(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~auth()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::ComboBox^ comboBox1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::PictureBox^ pictureBox1;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(auth::typeid));

this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

this->SuspendLayout();

//

// comboBox1

//

this->comboBox1->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 10, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(3) { L"Т-444", L"Т-445", L"Т-446" });

this->comboBox1->Location = System::Drawing::Point(118, 307);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(121, 24);

this->comboBox1->TabIndex = 0;

//

// textBox1

//

this->textBox1->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->textBox1->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->ForeColor = System::Drawing::SystemColors::Desktop;

this->textBox1->Location = System::Drawing::Point(118, 273);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(121, 26);

this->textBox1->TabIndex = 1;

//

// textBox2

//

this->textBox2->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->textBox2->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox2->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox2->ForeColor = System::Drawing::SystemColors::Desktop;

this->textBox2->Location = System::Drawing::Point(118, 340);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(121, 26);

this->textBox2->TabIndex = 2;

//

// button1

//

this->button1->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->button1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 14, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->Location = System::Drawing::Point(133, 386);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(87, 40);

this->button1->TabIndex = 3;

this->button1->Text = L"ВОЙТИ";

this->button1->UseVisualStyleBackColor = false;

this->button1->Click += gcnew System::EventHandler(this, &auth::button1\_Click);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(55, 272);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(58, 19);

this->label1->TabIndex = 4;

this->label1->Text = L"Логин:";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(51, 306);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(64, 19);

this->label2->TabIndex = 5;

this->label2->Text = L"Группа:";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(45, 340);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(68, 19);

this->label3->TabIndex = 6;

this->label3->Text = L"Пароль:";

//

// pictureBox1

//

this->pictureBox1->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox1.BackgroundImage")));

this->pictureBox1->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->pictureBox1->Location = System::Drawing::Point(81, 24);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(203, 211);

this->pictureBox1->TabIndex = 7;

this->pictureBox1->TabStop = false;

//

// auth

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::LightSkyBlue;

this->ClientSize = System::Drawing::Size(358, 489);

this->Controls->Add(this->pictureBox1);

this->Controls->Add(this->label3);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->button1);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->comboBox1);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->Name = L"auth";

this->Text = L"163umneck";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

try {

// Сбор логина и пароля с txt

String^ name = "login.txt";

StreamReader^ file = File::OpenText(name);

String^ log = file->ReadToEnd();

String^ l = Convert::ToString(this->textBox1->Text);

String^ p = Convert::ToString(this->textBox2->Text);

String^ namep = "password.txt";

StreamReader^ ffile = File::OpenText(namep);

String^ pas = ffile->ReadToEnd();

// Проверка введённых данных

if (log == l && p == pas && comboBox1->Text == "Т-444") {

StreamWriter^ group = gcnew StreamWriter("current.txt");

group->WriteLine("Т-444");

group->Close();

MessageBox::Show("Вы успешно вошли в аккаунт!");

labs^ form = gcnew labs;

this->Hide();

form->Show();

}

// Проверка введённых данных

else if (log == l && p == pas && comboBox1->Text == "Т-445") {

StreamWriter^ group = gcnew StreamWriter("current.txt");

group->WriteLine("Т-445");

group->Close();

MessageBox::Show("Вы успешно вошли в аккаунт!");

labs^ form = gcnew labs;

this->Hide();

form->Show();

}

// Проверка введённых данных

else if (log == l && p == pas && comboBox1->Text == "Т-446") {

StreamWriter^ group = gcnew StreamWriter("current.txt");

group->WriteLine("Т-446");

group->Close();

MessageBox::Show("Вы успешно вошли в аккаунт!");

labs^ form = gcnew labs;

this->Hide();

form->Show();

}

else {

MessageBox::Show("Проверьте введенные данные!");

this->textBox1->Clear();

this->comboBox1->Text = "";

this->textBox2->Clear();

}

}

catch (FormatException^ e) {

MessageBox::Show("Ошибка при авторизации");

}

}

};

}

«labs.h»

#pragma once

#include <cliext/map>

namespace kurs {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Collections::Generic;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::IO;

/// <summary>

/// Сводка для labs

/// </summary>

public ref class labs : public System::Windows::Forms::Form

{

public:

labs(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~labs()

{

if (components)

{

delete components;

}

}

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container^ components;

private: System::Windows::Forms::DataGridView^ dataGridView1;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::TextBox^ textBox5;

private: System::Windows::Forms::TextBox^ textBox6;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::ComboBox^ comboBox1;

private: System::Windows::Forms::ComboBox^ comboBox2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::Button^ button5;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::Button^ button6;

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::Button^ button7;

private: System::Windows::Forms::Label^ label6;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(labs::typeid));

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->button1 = (gcnew System::Windows::Forms::Button());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->button3 = (gcnew System::Windows::Forms::Button());

this->button4 = (gcnew System::Windows::Forms::Button());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->button2 = (gcnew System::Windows::Forms::Button());

this->textBox5 = (gcnew System::Windows::Forms::TextBox());

this->textBox6 = (gcnew System::Windows::Forms::TextBox());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label6 = (gcnew System::Windows::Forms::Label());

this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

this->comboBox2 = (gcnew System::Windows::Forms::ComboBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

this->button5 = (gcnew System::Windows::Forms::Button());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->button6 = (gcnew System::Windows::Forms::Button());

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->button7 = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

this->SuspendLayout();

//

// dataGridView1

//

this->dataGridView1->AllowUserToAddRows = false;

this->dataGridView1->AllowUserToDeleteRows = false;

this->dataGridView1->BackgroundColor = System::Drawing::SystemColors::ActiveCaption;

this->dataGridView1->ColumnHeadersBorderStyle = System::Windows::Forms::DataGridViewHeaderBorderStyle::Single;

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Location = System::Drawing::Point(24, 23);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->RowHeadersWidth = 51;

this->dataGridView1->Size = System::Drawing::Size(406, 171);

this->dataGridView1->TabIndex = 0;

//

// button1

//

this->button1->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->button1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->Location = System::Drawing::Point(596, 120);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(115, 31);

this->button1->TabIndex = 1;

this->button1->Text = L"Добавить ЛР";

this->button1->UseVisualStyleBackColor = false;

this->button1->Click += gcnew System::EventHandler(this, &labs::button1\_Click);

//

// textBox1

//

this->textBox1->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->textBox1->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->ForeColor = System::Drawing::SystemColors::Desktop;

this->textBox1->Location = System::Drawing::Point(611, 40);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(100, 26);

this->textBox1->TabIndex = 2;

//

// textBox2

//

this->textBox2->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->textBox2->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox2->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox2->ForeColor = System::Drawing::SystemColors::Desktop;

this->textBox2->Location = System::Drawing::Point(611, 84);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(100, 26);

this->textBox2->TabIndex = 3;

//

// button3

//

this->button3->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->button3->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 14, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button3->Location = System::Drawing::Point(312, 209);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(118, 32);

this->button3->TabIndex = 6;

this->button3->Text = L"Сохранить";

this->button3->UseVisualStyleBackColor = false;

this->button3->Click += gcnew System::EventHandler(this, &labs::button3\_Click);

//

// button4

//

this->button4->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->button4->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button4->Location = System::Drawing::Point(776, 121);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(100, 31);

this->button4->TabIndex = 7;

this->button4->Text = L"Удалить ЛР";

this->button4->UseVisualStyleBackColor = false;

this->button4->Click += gcnew System::EventHandler(this, &labs::button4\_Click);

//

// textBox4

//

this->textBox4->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->textBox4->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox4->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox4->ForeColor = System::Drawing::SystemColors::Desktop;

this->textBox4->Location = System::Drawing::Point(776, 84);

this->textBox4->Name = L"textBox4";

this->textBox4->Size = System::Drawing::Size(100, 26);

this->textBox4->TabIndex = 8;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(517, 42);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(77, 18);

this->label1->TabIndex = 9;

this->label1->Text = L"Номер ЛР:";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(473, 86);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(121, 18);

this->label2->TabIndex = 10;

this->label2->Text = L"Дата начала ЛР:\r\n";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label4->Location = System::Drawing::Point(787, 59);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(75, 18);

this->label4->TabIndex = 12;

this->label4->Text = L"Номер ЛР";

//

// button2

//

this->button2->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->button2->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button2->Location = System::Drawing::Point(302, 347);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(83, 59);

this->button2->TabIndex = 13;

this->button2->Text = L"Экспорт отчета";

this->button2->UseVisualStyleBackColor = false;

this->button2->Click += gcnew System::EventHandler(this, &labs::button2\_Click\_1);

//

// textBox5

//

this->textBox5->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->textBox5->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox5->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox5->ForeColor = System::Drawing::SystemColors::Desktop;

this->textBox5->Location = System::Drawing::Point(31, 360);

this->textBox5->Name = L"textBox5";

this->textBox5->Size = System::Drawing::Size(100, 26);

this->textBox5->TabIndex = 14;

//

// textBox6

//

this->textBox6->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->textBox6->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox6->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox6->ForeColor = System::Drawing::SystemColors::Desktop;

this->textBox6->Location = System::Drawing::Point(183, 360);

this->textBox6->Name = L"textBox6";

this->textBox6->Size = System::Drawing::Size(100, 26);

this->textBox6->TabIndex = 15;

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label5->Location = System::Drawing::Point(11, 364);

this->label5->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(15, 18);

this->label5->TabIndex = 16;

this->label5->Text = L"с";

//

// label6

//

this->label6->AutoSize = true;

this->label6->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label6->Location = System::Drawing::Point(153, 364);

this->label6->Margin = System::Windows::Forms::Padding(2, 0, 2, 0);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(25, 18);

this->label6->TabIndex = 17;

this->label6->Text = L"по";

//

// comboBox1

//

this->comboBox1->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 10, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(2) { L"лабораторные работы", L"фамилии" });

this->comboBox1->Location = System::Drawing::Point(135, 214);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(160, 24);

this->comboBox1->TabIndex = 18;

//

// comboBox2

//

this->comboBox2->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;

this->comboBox2->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 10, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->comboBox2->FormattingEnabled = true;

this->comboBox2->Items->AddRange(gcnew cli::array< System::Object^ >(2) { L"возрастанию", L"убыванию" });

this->comboBox2->Location = System::Drawing::Point(135, 254);

this->comboBox2->Name = L"comboBox2";

this->comboBox2->Size = System::Drawing::Size(160, 24);

this->comboBox2->TabIndex = 19;

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(23, 213);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(99, 18);

this->label3->TabIndex = 20;

this->label3->Text = L"Сортировать:";

//

// label7

//

this->label7->AutoSize = true;

this->label7->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label7->Location = System::Drawing::Point(97, 254);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(25, 18);

this->label7->TabIndex = 21;

this->label7->Text = L"по";

//

// button5

//

this->button5->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->button5->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button5->Location = System::Drawing::Point(149, 292);

this->button5->Name = L"button5";

this->button5->Size = System::Drawing::Size(118, 35);

this->button5->TabIndex = 22;

this->button5->Text = L"Сортировать";

this->button5->UseVisualStyleBackColor = false;

this->button5->Click += gcnew System::EventHandler(this, &labs::button5\_Click);

//

// textBox3

//

this->textBox3->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->textBox3->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->textBox3->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 11.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox3->ForeColor = System::Drawing::SystemColors::Desktop;

this->textBox3->Location = System::Drawing::Point(499, 219);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(137, 26);

this->textBox3->TabIndex = 23;

//

// button6

//

this->button6->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->button6->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button6->Location = System::Drawing::Point(530, 254);

this->button6->Name = L"button6";

this->button6->Size = System::Drawing::Size(78, 41);

this->button6->TabIndex = 24;

this->button6->Text = L"Искать";

this->button6->UseVisualStyleBackColor = false;

this->button6->Click += gcnew System::EventHandler(this, &labs::button6\_Click);

//

// richTextBox1

//

this->richTextBox1->BackColor = System::Drawing::SystemColors::ActiveCaption;

this->richTextBox1->Font = (gcnew System::Drawing::Font(L"Bahnschrift", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->richTextBox1->Location = System::Drawing::Point(667, 179);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->ReadOnly = true;

this->richTextBox1->Size = System::Drawing::Size(209, 140);

this->richTextBox1->TabIndex = 25;

this->richTextBox1->Text = L"";

//

// button7

//

this->button7->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"button7.BackgroundImage")));

this->button7->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Zoom;

this->button7->Location = System::Drawing::Point(806, 338);

this->button7->Name = L"button7";

this->button7->Size = System::Drawing::Size(70, 68);

this->button7->TabIndex = 26;

this->button7->UseVisualStyleBackColor = true;

this->button7->Click += gcnew System::EventHandler(this, &labs::button7\_Click);

//

// labs

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::LightSkyBlue;

this->ClientSize = System::Drawing::Size(905, 426);

this->Controls->Add(this->button7);

this->Controls->Add(this->richTextBox1);

this->Controls->Add(this->button6);

this->Controls->Add(this->textBox3);

this->Controls->Add(this->button5);

this->Controls->Add(this->label7);

this->Controls->Add(this->label3);

this->Controls->Add(this->comboBox2);

this->Controls->Add(this->comboBox1);

this->Controls->Add(this->label6);

this->Controls->Add(this->label5);

this->Controls->Add(this->textBox6);

this->Controls->Add(this->textBox5);

this->Controls->Add(this->button2);

this->Controls->Add(this->label4);

this->Controls->Add(this->label2);

this->Controls->Add(this->label1);

this->Controls->Add(this->textBox4);

this->Controls->Add(this->button4);

this->Controls->Add(this->button3);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->button1);

this->Controls->Add(this->dataGridView1);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->KeyPreview = true;

this->Name = L"labs";

this->Text = L"163umneck";

this->Load += gcnew System::EventHandler(this, &labs::labs\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

private :

// Функция для проверки даты и окрашивания ячеек

void CheckDatesAndColorCells(System::Windows::Forms::DataGridView^ dataGridView) {

// Получите текущую дату

System::DateTime currentDate = System::DateTime::Now.Date;

// Просмотрите все столбцы

for (int i = 0; i < dataGridView->Columns->Count; ++i) {

// Получите название столбца

System::String^ columnName = dataGridView->Columns[i]->Name;

// Проверьте, содержит ли название столбца дату последней сдачи

if (columnName->Contains("(") && columnName->Contains(")")) {

// Извлеките дату последней сдачи из названия столбца

int startIndex = columnName->IndexOf("(") + 1;

int endIndex = columnName->IndexOf(")");

System::String^ endDateStr = columnName->Substring(startIndex, endIndex - startIndex);

// Преобразуйте строку в дату

System::DateTime endDate = System::DateTime::Parse(endDateStr);

// Если текущая дата превышает дату последней сдачи, окрасьте ячейки в красный цвет

// Если текущая дата равна дате последней сдачи, окрасьте ячейки в желтый цвет

// Если дата последней сдачи еще не наступила, окрасьте ячейки в белый цвет

for each (System::Windows::Forms::DataGridViewRow ^ row in dataGridView->Rows) {

// Проверьте, является ли ячейка пустой

if (!row->IsNewRow && (row->Cells[i]->Value == "" || row->Cells[i]->Value == nullptr)) {

if (currentDate > endDate) {

row->Cells[i]->Style->BackColor = System::Drawing::Color::Red;

}

else if (currentDate == endDate) {

row->Cells[i]->Style->BackColor = System::Drawing::Color::Yellow;

}

else if (currentDate < endDate) {

row->Cells[i]->Style->BackColor = System::Drawing::Color::White;

}

}

}

}

}

}

// Функция для сохранения данных из DataGridView в файл CSV

void SaveDataToCSV(System::Windows::Forms::DataGridView^ dataGridView, String^ filePath) {

StreamWriter^ sw = gcnew StreamWriter(filePath, false, System::Text::Encoding::Default);

String^ data = "";

// Запись заголовков столбцов

for each (DataGridViewColumn ^ column in dataGridView->Columns) {

data += column->HeaderText + ";";

}

sw->WriteLine(data);

// Запись данных

for each (DataGridViewRow ^ row in dataGridView->Rows) {

data = "";

for each (DataGridViewCell ^ cell in row->Cells) {

data += cell->Value + ";";

}

sw->WriteLine(data);

}

sw->Close();

}

// Функция для загрузки данных из файла CSV в DataGridView

void LoadDataFromCSV(System::Windows::Forms::DataGridView^ dataGridView, String^ filePath) {

StreamReader^ sr = gcnew StreamReader(filePath, System::Text::Encoding::Default);

String^ line;

// Чтение данных

line = sr->ReadLine();

if (line != nullptr) {

// Создание столбцов

array<String^>^ headers = line->Split(';');

for each (String ^ header in headers) {

dataGridView->Columns->Add(header, header);

}

// Чтение и добавление строк

while ((line = sr->ReadLine()) != nullptr) {

array<String^>^ items = line->Split(';');

dataGridView->Rows->Add(items);

}

for (int i = 0; i < dataGridView1->Columns->Count; i++)

{

if (String::IsNullOrEmpty(dataGridView1->Columns[i]->HeaderText))

{

dataGridView1->Columns->RemoveAt(i);

i--; // Уменьшаем индекс, т.к. столбец был удален

}

}

}

sr->Close();

}

#pragma endregion

private: System::Void labs\_Load(System::Object^ sender, System::EventArgs^ e) {

StreamReader^ group = File::OpenText("current.txt");

String^ name = group->ReadToEnd();

group->Close();

name = name->Trim();

String^ txt = Convert::ToString(name) + ".csv";

LoadDataFromCSV(dataGridView1, txt);

CheckDatesAndColorCells(dataGridView1);

dataGridView1->CellValueChanged += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &labs::dataGridView1\_CellValueChanged);

FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this, &labs::labs\_FormClosing);

ToolTip^ toolTip1 = gcnew ToolTip;

toolTip1->SetToolTip(this->textBox1, "В это поле можно вводить целые положительные числа");

ToolTip^ toolTip2 = gcnew ToolTip;

toolTip2->SetToolTip(this->textBox2, "В это поле вводится дата в формате день.месяц");

ToolTip^ toolTip3 = gcnew ToolTip;

toolTip3->SetToolTip(this->textBox3, "В это поле можно вводить фамилии учащихся для поиска по фамилии\r\nи 'ЛР'/'Лабораторная работа ' номер для поиска по ЛР");

ToolTip^ toolTip4 = gcnew ToolTip;

toolTip4->SetToolTip(this->textBox4, "В это поле можно вводить целые положительные числа");

ToolTip^ toolTip5 = gcnew ToolTip;

toolTip5->SetToolTip(this->textBox5, "В это поле вводится дата в формате день.месяц");

ToolTip^ toolTip6 = gcnew ToolTip;

toolTip6->SetToolTip(this->textBox6, "В это поле вводится дата в формате день.месяц");

}

private: System::Void labs\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {

Application::Exit();

}

private: System::Void dataGridView1\_CellValueChanged(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e) {

// Обработчик событий для изменения значений ячеек

System::Windows::Forms::DataGridView^ dataGridView = safe\_cast<System::Windows::Forms::DataGridView^>(sender);

if (e->RowIndex >= 0 && e->ColumnIndex >= 0) {

System::Windows::Forms::DataGridViewCell^ cell = dataGridView->Rows[e->RowIndex]->Cells[e->ColumnIndex];

if (cell->Value != nullptr && cell->Value->ToString() != "") {

try {

int value = Convert::ToInt32(cell->Value);

if (value >= 1 && value <= 10) {

cell->Tag = cell->Value; // Сохраняем новое значение в Tag

}

else {

// Недопустимое значение, возвращаем старое и выводим предупреждение

System::Windows::Forms::MessageBox::Show("Пожалуйста, введите целое число от 1 до 10.", "Неверное значение", System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Warning);

cell->Value = cell->Tag != nullptr ? cell->Tag->ToString() : "";

}

}

catch (FormatException^) {

// Недопустимое значение, возвращаем старое и выводим предупреждение

System::Windows::Forms::MessageBox::Show("Пожалуйста, введите целое число от 1 до 10.", "Неверное значение", System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Warning);

cell->Value = cell->Tag != nullptr ? cell->Tag->ToString() : "";

}

}

else {

// Если значение пустое, сохраняем его в Tag и не показываем предупреждение

cell->Tag = "";

}

// Если значение пустое, сохраняем пустое значение в Tag

if (cell->Value == nullptr || cell->Value->ToString() == "") {

cell->Tag = "";

}

// Если ячейка была красной и пустой, то поменять её цвет на белый при изменении значения

if (cell->Style->BackColor == System::Drawing::Color::Red && cell->Value != "" && cell->Value != nullptr) {

cell->Style->BackColor = System::Drawing::Color::White;

}

}

// Применяем проверку дат и окраску ячеек

CheckDatesAndColorCells(dataGridView);

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

try {

if (textBox1->Text != "") {

String^ labNumber = textBox1->Text;

System::Text::RegularExpressions::Regex^ integerRegex = gcnew System::Text::RegularExpressions::Regex("^[0-9]+$");

if (!integerRegex->IsMatch(labNumber)) {

MessageBox::Show("В поле 'Номер ЛР' можно вводить только целые числа");

textBox1->Clear();

return;

}

// Проверка на существование номера лабораторной работы

for (int i = 0; i < dataGridView1->Columns->Count; ++i) {

if (dataGridView1->Columns[i]->Name->Contains("ЛР" + labNumber)) {

MessageBox::Show("Такой номер лабораторной работы уже существует");

textBox1->Clear();

return;

; }

}

if (textBox2->Text != "") {

String^ startDate = textBox2->Text;

// Проверка формата даты

System::Text::RegularExpressions::Regex^ dateRegex = gcnew System::Text::RegularExpressions::Regex("^\\d{2}\\.\\d{2}$");

if (!dateRegex->IsMatch(startDate)) {

MessageBox::Show("Дата начала ЛР введена в неправильном формате. Формат должен быть 01.01 (число.месяц)");

textBox2->Clear();

return;

}

// Расчет конечной даты

array<String^>^ dateParts = startDate->Split('.');

int day = Int32::Parse(dateParts[0]);

int month = Int32::Parse(dateParts[1]);

DateTime startDateTime = DateTime(DateTime::Now.Year, month, day);

DateTime endDateTime = startDateTime.AddDays(14);

String^ endDate = endDateTime.ToString("dd.MM");

String^ columnName = "ЛР" + labNumber + " " + startDate + " (" + endDate + ")";

dataGridView1->Columns->Add(columnName, columnName);

// Заполните новый столбец пустыми значениями

for each (DataGridViewRow ^ row in dataGridView1->Rows) {

row->Cells[columnName]->Value = "";

}

CheckDatesAndColorCells(dataGridView1);

StreamReader^ group = File::OpenText("current.txt");

String^ name = group->ReadToEnd();

group->Close();

name = name->Trim();

String^ tex = Convert::ToString(name) + ".csv";

SaveDataToCSV(dataGridView1, tex);

}

else {

MessageBox::Show("Введите дату начала ЛР");

}

}

else {

MessageBox::Show("Введите номер лабораторной работы");

}

}

catch (Exception^ ex) {

MessageBox::Show("Введена некорректная дата");

textBox2->Clear();

}

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

StreamReader^ group = File::OpenText("current.txt");

String^ name = group->ReadToEnd();

group->Close();

name = name->Trim();

String^ tex = Convert::ToString(name) + ".csv";

SaveDataToCSV(dataGridView1,tex);

CheckDatesAndColorCells(dataGridView1);

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

if (textBox4->Text != "") {

String^ labNumber = textBox4->Text;

System::Text::RegularExpressions::Regex^ integerRegex = gcnew System::Text::RegularExpressions::Regex("^[0-9]+$");

if (!integerRegex->IsMatch(labNumber)) {

MessageBox::Show("В поле 'Номер ЛР' можно вводить только целые числа");

textBox4->Clear();

return;

}

if (textBox4->Text != "" && (Convert::ToInt32(textBox4->Text) > 0)) {

String^ labNumber = textBox4->Text;

int columnIndex = -1;

// Проверка на существование номера лабораторной работы

for (int i = 0; i < dataGridView1->Columns->Count; ++i) {

if (dataGridView1->Columns[i]->Name->Contains("ЛР" + labNumber)) {

columnIndex = i;

break;

}

}

if (columnIndex != -1) {

dataGridView1->Columns->RemoveAt(columnIndex);

}

else {

MessageBox::Show("Лабораторная работа с номером " + labNumber + " не найдена.");

}

}

else {

MessageBox::Show("Введите корректный номер лабораторной работы для удаления");

this->textBox4->Clear();

}

}

else {

MessageBox::Show("Введите номер лабораторной работы");

}

}

private: System::Void button2\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

// Чтение дат из текстбоксов

if (textBox5->Text != "") {

if (textBox6->Text != "") {

String^ startDateStr = textBox5->Text;

String^ endDateStr = textBox6->Text;

// Проверка формата дат с использованием регулярных выражений

System::Text::RegularExpressions::Regex^ dateRegex = gcnew System::Text::RegularExpressions::Regex("^\\d{2}\\.\\d{2}$");

if (!dateRegex->IsMatch(startDateStr)) {

MessageBox::Show("Дата начала введена в неправильном формате. Формат должен быть 01.01 (число.месяц)");

textBox5->Clear();

return;

}

if (!dateRegex->IsMatch(endDateStr)) {

MessageBox::Show("Дата окончания введена в неправильном формате. Формат должен быть 01.01 (число.месяц)");

textBox6->Clear();

return;

}

// Преобразование строковых дат в DateTime, добавление текущего года для обработки даты

DateTime startDate = DateTime::ParseExact(startDateStr + ".2024", "dd.MM.yyyy", nullptr);

DateTime endDate = DateTime::ParseExact(endDateStr + ".2024", "dd.MM.yyyy", nullptr);

String^ report = "";

// Перебор всех столбцов, начиная со второго, поскольку первый столбец содержит фамилии

for (int colIndex = 1; colIndex < dataGridView1->Columns->Count; ++colIndex) {

String^ columnName = dataGridView1->Columns[colIndex]->Name;

// Извлечение и проверка даты окончания из названия столбца

if (columnName->Contains("ЛР") && columnName->Contains("(") && columnName->Contains(")")) {

int endDateIndex = columnName->IndexOf("(") + 1;

String^ endDateStrColumn = columnName->Substring(endDateIndex, columnName->IndexOf(")") - endDateIndex);

DateTime labEndDate = DateTime::ParseExact(endDateStrColumn + ".2024", "dd.MM.yyyy", nullptr);

// Проверка, находится ли дата окончания в заданном диапазоне

if (labEndDate >= startDate && labEndDate <= endDate) {

// Добавление абзацов для разделения лабораторных работ

if (!String::IsNullOrEmpty(report)) {

report += "\r\n";

}

report += columnName->Substring(0, columnName->IndexOf(" ")) + ":\r\n";

String^ notSubmitted = "Не сдали: ";

bool hasNonSubmitted = false;

bool hasSubmitted = false;

for each (System::Windows::Forms::DataGridViewRow ^ row in dataGridView1->Rows) {

if (row->IsNewRow) continue;

String^ studentName = row->Cells[0]->Value->ToString(); // Фамилия ученика

auto cellValue = row->Cells[colIndex]->Value; // Оценка за лабораторную работу

if (cellValue != nullptr && cellValue->ToString() != "") {

report += studentName + " " + cellValue->ToString() + ". ";

hasSubmitted = true;

}

else {

hasNonSubmitted = true;

notSubmitted += studentName + ", ";

}

}

// Удаление лишней запятой и пробела в конце строки для "Не сдали"

if (hasNonSubmitted) {

notSubmitted = notSubmitted->TrimEnd(gcnew array<wchar\_t> { ',', ' ' });

report += "\r\n" + notSubmitted;

}

report += "\r\n";

}

}

}

report = report->TrimEnd();

SaveFileDialog^ saveFileDialog1 = gcnew SaveFileDialog();

saveFileDialog1->Title = "Сохранить отчет";

saveFileDialog1->DefaultExt = "txt";

saveFileDialog1->Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

saveFileDialog1->RestoreDirectory = true;

if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {

StreamWriter^ sw = gcnew StreamWriter(saveFileDialog1->FileName, false, System::Text::Encoding::Default);

sw->WriteLine(report);

sw->Close();

System::Windows::Forms::MessageBox::Show("Отчет был создан", "", System::Windows::Forms::MessageBoxButtons::OK, System::Windows::Forms::MessageBoxIcon::Information);

}

}

else {

MessageBox::Show("Введите конечную дату");

}

}

else {

MessageBox::Show("Введите начальную дату");

}

}

// Класс компаратора для сортировки столбцов

ref class ColumnComparer : public System::Collections::Generic::IComparer<DataGridViewColumn^> {

public:

String^ sortOrder;

ColumnComparer(String^ sortOrder) {

this->sortOrder = sortOrder;

}

virtual int Compare(DataGridViewColumn^ a, DataGridViewColumn^ b) {

int aNumber = Int32::Parse(a->Name->Substring(2, a->Name->IndexOf(" ") - 2));

int bNumber = Int32::Parse(b->Name->Substring(2, b->Name->IndexOf(" ") - 2));

return sortOrder == "возрастанию" ? aNumber.CompareTo(bNumber) : bNumber.CompareTo(aNumber);

}

};

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

// Получаем выбранные критерии сортировки

if (comboBox1->Text != "") {

if (comboBox2->Text != "") {

String^ sortField = comboBox1->SelectedItem->ToString();

String^ sortOrder = comboBox2->SelectedItem->ToString();

// Проверяем, какой критерий сортировки был выбран

if (sortField == "фамилии") {

// Сортируем по фамилии

dataGridView1->Sort(dataGridView1->Columns["Фамилия"], sortOrder == "возрастанию" ? ListSortDirection::Ascending : ListSortDirection::Descending);

}

else if (sortField == "лабораторные работы") {

// Сортируем столбцы по номеру лабораторной работы

List<DataGridViewColumn^>^ columns = gcnew List<DataGridViewColumn^>();

for (int i = 1; i < dataGridView1->Columns->Count; i++) {

DataGridViewColumn^ column = dataGridView1->Columns[i];

if (column->Name->StartsWith("ЛР")) {

columns->Add(column);

}

}

columns->Sort(gcnew ColumnComparer(sortOrder));

for (int i = 0; i < columns->Count; i++) {

columns[i]->DisplayIndex = i + 1;

}

}

}

else {

MessageBox::Show("Введите критерий сортировки");

}

}

else {

MessageBox::Show("Введите параметр сортировки");

}

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

if (textBox3->Text != "") {

// Получаем введенное пользователем значение для поиска

String^ searchValue = textBox3->Text;

try {

// Очищаем richtextbox1 перед выводом результатов

richTextBox1->Clear();

if (searchValue->StartsWith("ЛР")) {

// Поиск по номеру лабораторной работы

String^ labNumberStr = searchValue->Substring(2)->Trim(); // Изменено здесь

int labNumber = Int32::Parse(labNumberStr); // Изменено здесь

String^ labColumn = "ЛР" + labNumber;

String^ passed = "Сдали: ";

String^ notPassed = "Не сдали: ";

bool found = false;

for (int i = 0; i < dataGridView1->Rows->Count; i++) {

DataGridViewRow^ row = dataGridView1->Rows[i];

for (int j = 1; j < dataGridView1->Columns->Count; j++) {

DataGridViewColumn^ column = dataGridView1->Columns[j];

if (column->Name->StartsWith(labColumn)) {

String^ cellValue = row->Cells[column->Name]->Value != nullptr ? row->Cells[column->Name]->Value->ToString() : "";

found = true;

if (!String::IsNullOrEmpty(cellValue)) {

passed += row->Cells["Фамилия"]->Value->ToString() + " (" + cellValue + "), ";

}

else {

notPassed += row->Cells["Фамилия"]->Value->ToString() + ", ";

}

}

}

}

if (found) {

richTextBox1->Text = "Лабораторная работа " + labNumber + ":\n" + passed->TrimEnd(',', ' ') + "\n" + notPassed->TrimEnd(',', ' ');

}

else {

MessageBox::Show("Не найдено результата по запросу " + searchValue);

}

}

else {

// Поиск по фамилии

String^ passed = "Сдал: ";

String^ notPassed = "Не сдал: ";

bool found = false;

for (int i = 0; i < dataGridView1->Rows->Count; i++) {

DataGridViewRow^ row = dataGridView1->Rows[i];

if (row->Cells["Фамилия"]->Value->ToString()->Equals(searchValue, StringComparison::OrdinalIgnoreCase)) {

found = true;

for (int j = 1; j < dataGridView1->Columns->Count; j++) {

DataGridViewCell^ cell = row->Cells[j];

String^ cellValue = cell->Value != nullptr ? cell->Value->ToString() : "";

String^ labWork = cell->OwningColumn->Name->Substring(0, cell->OwningColumn->Name->IndexOf(' '));

if (!String::IsNullOrEmpty(cellValue)) {

passed += labWork + " (" + cellValue + "), ";

}

else {

notPassed += labWork + ", ";

}

}

break;

}

}

if (found) {

richTextBox1->Text = "Учащийся " + searchValue + ":\n" + passed->TrimEnd(',', ' ') + "\n" + notPassed->TrimEnd(',', ' ');

}

else {

MessageBox::Show("Не найдено результата по запросу " + searchValue);

}

}

}

catch (Exception^ ex) {

// Обрабатываем исключения и выводим сообщение об ошибке

MessageBox::Show("Не найдено результата по запросу " + searchValue);

}

}

else {

MessageBox::Show("Введите запрос для поиска");

}

}

private: System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ path = System::IO::Path::GetFullPath("Справочная система.pdf");

if (!System::IO::File::Exists(path)) return;

System::Diagnostics::Process::Start(path);

}

};

}

Приложение Б

**(справочное)**

**Результаты работы приложения**

На рисунках Б.1 - Б.10 представлены результаты тестирования приложения.

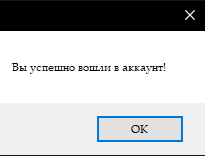


Рисунок Б.1 – Предупреждение об успешном входе

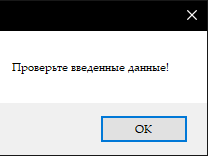


Рисунок Б.2 – Предупреждение об введении некорректных данных

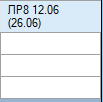


Рисунок Б.3 – Добавление лабораторной работы в таблицу

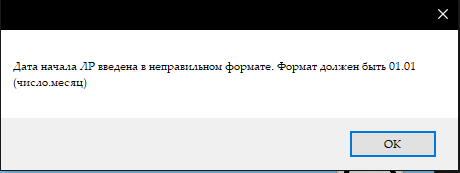


Рисунок Б.4 – Предупреждение об введении некорректных данных

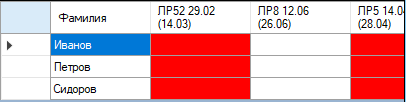


Рисунок Б.5 – Сортировка таблицы

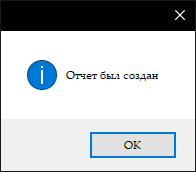


Рисунок Б.6 – Предупреждение об успешном создании отчета

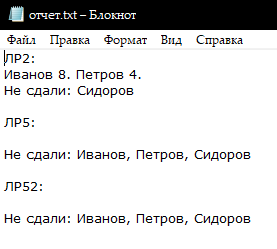


Рисунок Б.7 – Файл отчета

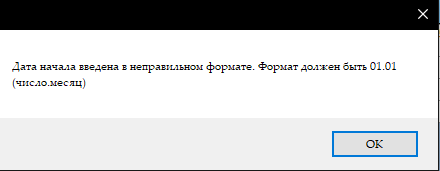


Рисунок Б.8 – Предупреждение об введении некорректных данных

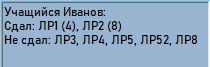


Рисунок Б.9 – Поиск данных из таблицы

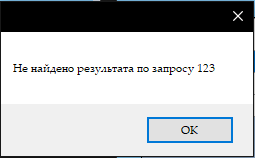


Рисунок Б.10 – Предупреждение об отсутствии результата