|  |
| --- |
| **Министерство образования и науки Российской Федерации**  Федеральное государственное автономное образовательное  учреждение высшего образования  **«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**  **ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»** |

Подразделение: Инженерная школа энергетики

Направление подготовки: 09.04.03 – Прикладная информатика

Отделение: Электроэнергетики и электротехники

**Проектная документация**

**Отчёт по лабораторной работе №5**

по дисциплине: «Основы объектно-ориентированного программирования»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент гр. О-5КМ11 | |  | \_\_\_\_\_\_\_\_\_\_ | |  | Братчиков Е.С. |
|  | |  |  | |  |  |
|  | |  |  |  | | |
|  | |  |  | |  |  |
| Отчёт принял | доцент, к.т.н. |  | \_\_\_\_\_\_\_\_\_\_ | |  | Калентьев А.А. |
|  |  |  |  | |  |  |
|  |  |  |  | \_\_\_\_\_\_\_\_\_ | | |

Томск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc154428216)

[1 Основная часть 4](#_Toc154428217)

[1.1 UML диаграмма вариантов использования 4](#_Toc154428218)

[1.3 Описание классов, образующих связь типа «общее-частное» 7](#_Toc154428219)

[1.5 Тестирование программы 10](#_Toc154428220)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc154428221)

**ВВЕДЕНИЕ**

Корректная и полная документация сопровождает разработку программного обеспечения (далее – ПО) от появления идеи до выпуска конечного продукта. Написание документации является обязательным критерием разработки и последующей поддержки проекта [1].

Целью данной лабораторной работы является разработка проектной документации на созданный программный продукт.

Для достижения поставленной цели должны быть выполнены следующие задачи:

* Составление технического задания (далее – ТЗ) на разработанную программу (Приложение А);
* Составление UML диаграммы вариантов использования для разработанной программы;
* Составление UML диаграммы классов;
* Описание классов, образующих связь типа «общее-частное»;
* Привести дерево ветвлений Git;
* Провести тестирование программы.

**1 Основная часть**

**1.1 UML диаграмма вариантов использования**

Вариант использования (use case) — это описание множества последовательных действий (включая вариации), которые выполняются некоторым субъектом с целью получения результата, значимого для некоторого действующего лица [1]. ВИ предполагает взаимодействие действующих лиц и системы или другого объекта. Действующее лицо представляет собой логически связанное множество ролей, которые играют пользователи системы во время взаимодействия с ней.

Диаграмма вариантов использования для разработанного ПО приведена на рисунке 1.

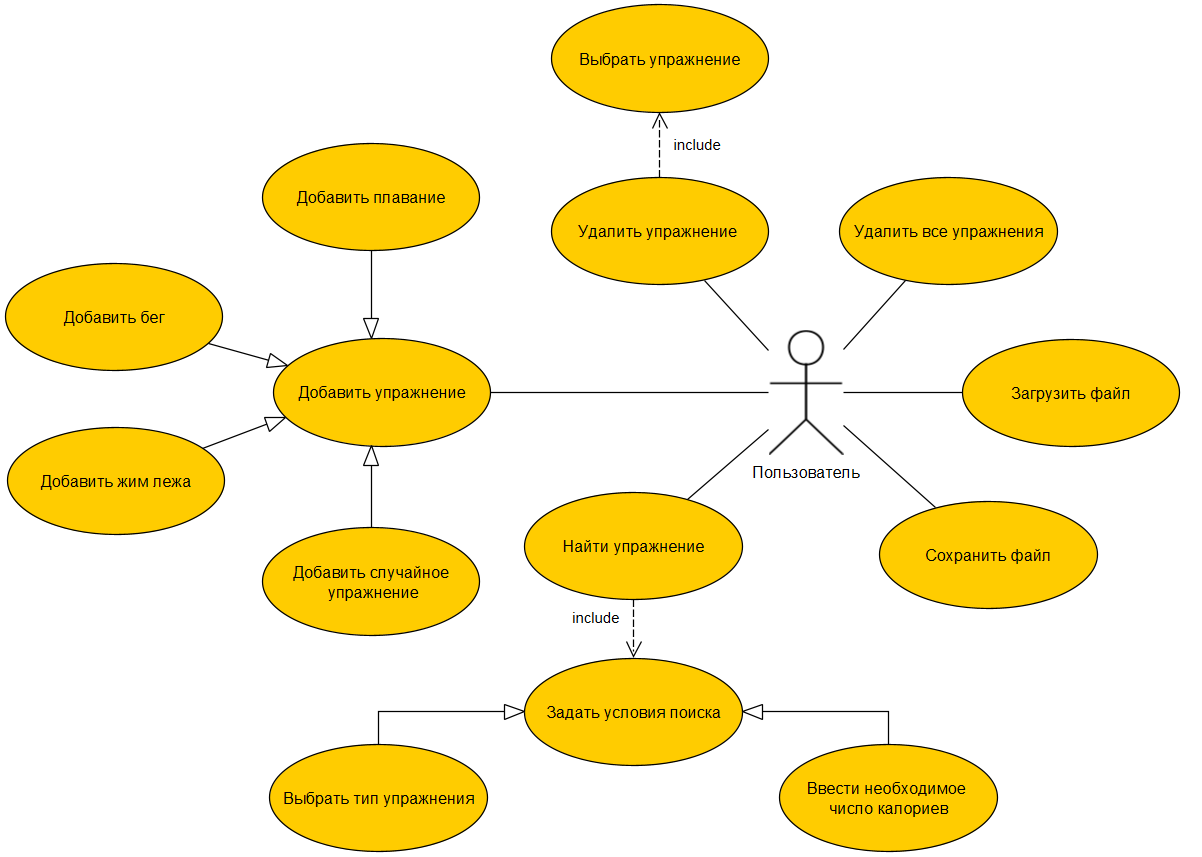


Рисунок 1 – Диаграмма вариантов использования

**1.2 UML диаграмма классов**

Диаграмма классов (англ. class diagram) — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма классов приведена на рисунке 2.

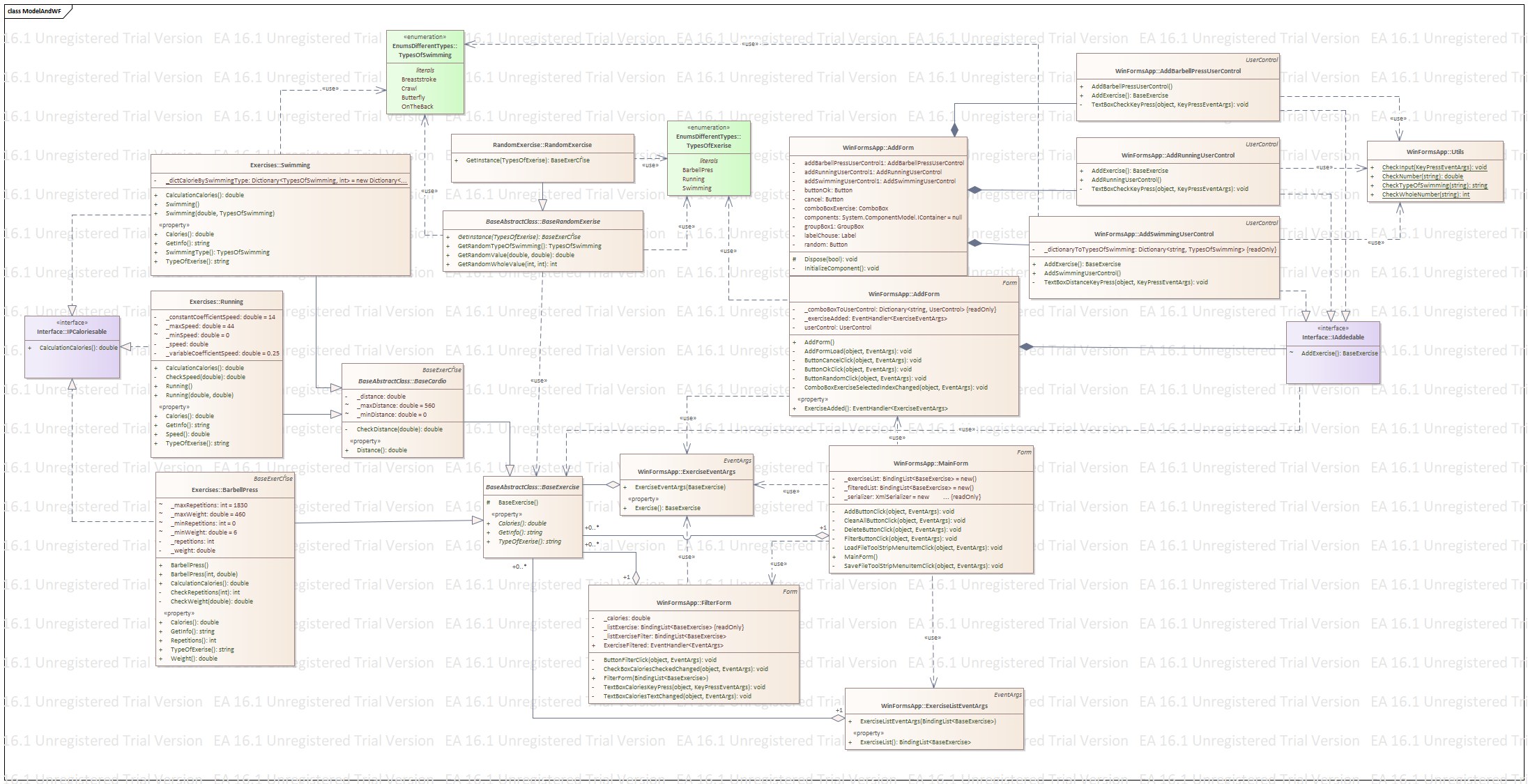


Рисунок 2 – UML диаграмма классов

**1.3 Описание классов, образующих связь типа «общее-частное»**

Описание абстрактных классов *BaseExercise, BaseRandomExerise, BaseCardio* с их свойствами и методами приведены в таблицах 1, 2 и 3 соответственно.

Таблица 1. Описание класса BaseExercise

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Упражнения (бег, плавание, жим штанги) | | |
| **Свойства** | | |
| + TypeOfExerise | string | Возвращает тип упражнения. |
| + GetInfo | string | Информация об упражнениях |
| + Calories | double | Возвращает информацию о сожжённых калориях |
| **Методы** | | |
| + BaseExerсise() |  | Конструктор класса |

Таблица 2. Описание класса BaseCardio

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Базовый класс кардио упражнений | | |
| **Свойства** | | |
| + Distance | string | Преодолённое расстояние |
| **Методы** | | |
| - CheckDistance |  | Метод проверяет заполнение расстония |

Таблица 3. Описание класса BaseRandomExerise

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Базовый класс случайного упражнения | | |
| **Методы** | | |
| + GetInstance | BaseExerсise | Получить экземпляр типа упражнения |
| +GetRandomTypeOfSwimming | TypesOfSwimming | Генератор рандомного типа плавания. |
| + GetRandomValue | double | Метод возвращает случайное число в указанном диапазоне. |
| + GetRandomWholeValue | int | Метод возвращает случайное целое число в указанном диапазоне. |

Таблица 4. Описание класса BarbellPress

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Описание жима штанги | | |
| **Свойства** | | |
| + Repetitions | int | Повторения |
| + Weight | double | Вес |
| + TypeOfExerise | string | Тип упражнения |
| + GetInfo | string | Информация по жиму штаги |
| + Calories | double | Число калориев |
| **Методы** | | |
| + CheckRepetitions | int | Метод проверяющий заполнение повторений. |
| + CheckWeight | double | Метод проверяющий заполнение веса. |
| + CalculationCalories | double | Вычисление расхода калориев при жиме штанги. |
| + BarbellPress() |  | Конструктор по умолчанию. |
| + BarbellPress(int repetitions, double weight) |  | Конструктор класса жим штанги. |

Таблица 5. Описание класса Running

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Описание бега | | |
| **Свойства** | | |
| + Speed | double | Скорость |
| + TypeOfExerise | string | Тип упражнения |
| + GetInfo | string | Информация по скорости бега |
| + Calories | double | Число калориев. |
| **Методы** | | |
| + CheckSpeed | double | Метод проверяющий заполнение скорости бега. |
| + CalculationCalories | double | Вычисление расхода калориев при беге. |
| + Running () |  | Конструктор по умолчанию. |
| + Running (double distance, double speed) |  | Конструктор класса бег. |

Таблица 6. Описание класса Running

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Описание плавания | | |
| **Свойства** | | |
| + SwimmingType | TypesOfSwimming | Скорость |
| + TypeOfExerise | string | Тип упражнения |
| + GetInfo | string | Информация по плаванию |
| + Calories | double | Число калориев. |
| **Методы** | | |
| + CalculationCalories | double | Вычисление расхода калориев при плавании. |
| + Swimming () |  | Конструктор по умолчанию. |
| + Swimming(double distance, TypesOfSwimming swimmingType) |  | Конструктор класса плавание. |

Таблица 7. Описание класса RandomExercise

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **Описание класса** | | |
| Описание создания рандомного упражнения | | |
| **Методы** | | |
| + GetInstance | BaseExerсise | Метод для получения случайного экземпляра упражнения. |

**1.4 Дерево ветвлений Git**

На рисунке 3 представлено дерево ветвлений Git, полученное по окончании работы с проектом.

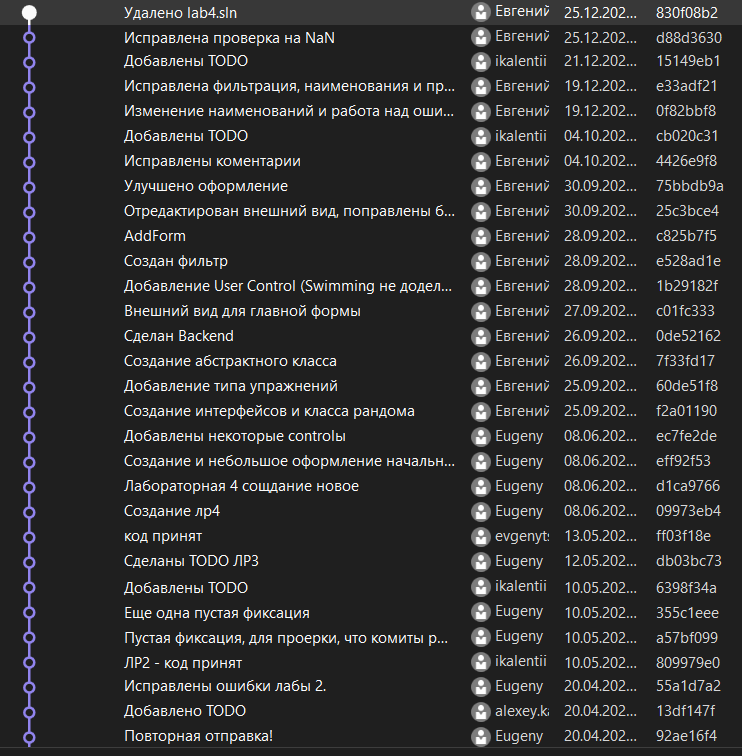


Рисунок 3 – Дерево ветвлений Git

**1.5 Тестирование программы**

Далее приводится процесс функционального тестирования программы.

Графический интерфейс пользователя представлен на рисунке 4.

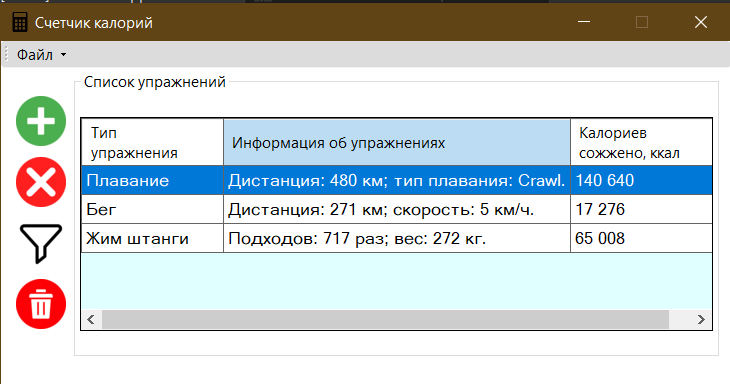


Рисунок 4 – Графический интерфейс пользователя

**1.5.1 Тестовый случай добавления элемента**

Для добавления элемента необходимо вызвать соответствующую форму путём нажатия кнопки «» (рисунок 5).

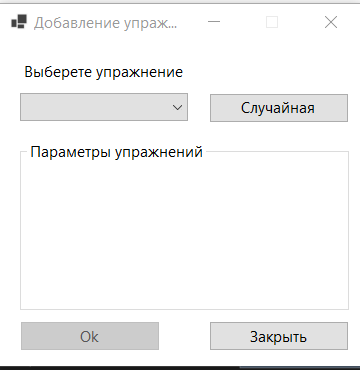
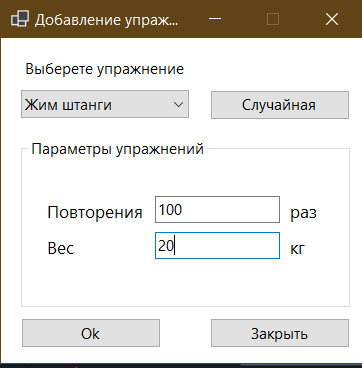
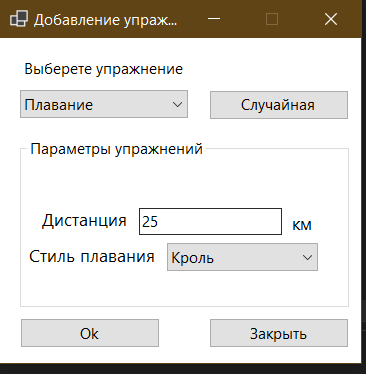


Рисунок 5 – Форма для добавления упражнения

Параметры упражнения можно указать в выпадающем меню.

После ввода данных необходимо нажать кнопку «Ок», элемент появится в таблице главной формы (рисунки 6 и 7).

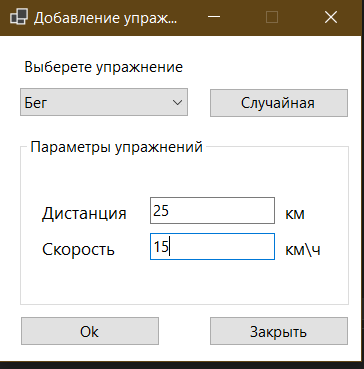


Рисунок 6 – Заполнение полей

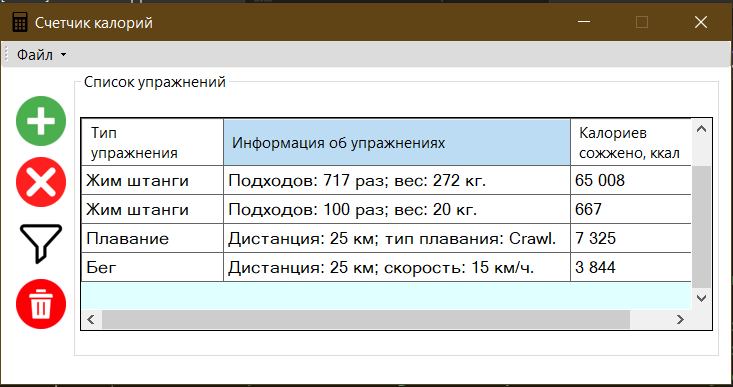
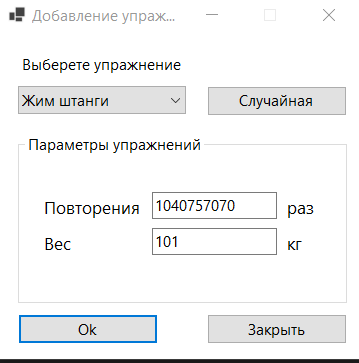


Рисунок 7 – Успешное добавление нового элемента

В программе предусмотрена система обработки некорректного ввода данных пользователем. Например, при попытке ввести число вне заранее определенного диапазона, появится соответствующее сообщение об ошибке (рисунки 8-9).



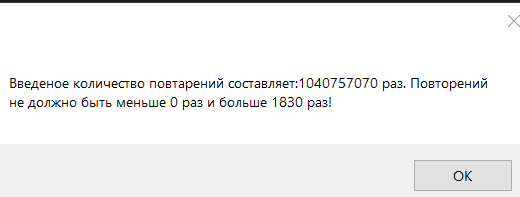
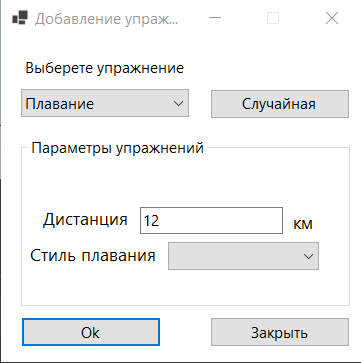


Рисунок 8 – Некорректный ввод (неверное число повторений)



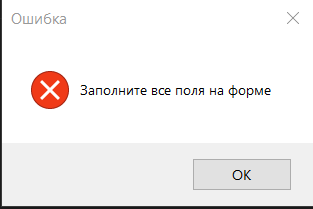


Рисунок 9 – Некорректный ввод (ввод недопустимых символов)

Подобная обработка предусмотрена для всех параметров элементов.

**1.5.2 Тестовый случай «Удалить элемент»**

Для удаления одного или нескольких изданий необходимо выбрать их в таблице и нажать на значок удаления (рисунки 10 и 11).

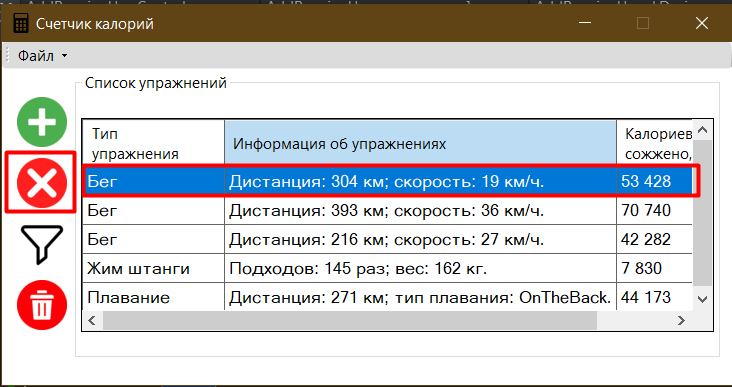


Рисунок 10 – Выбор элемента в таблице

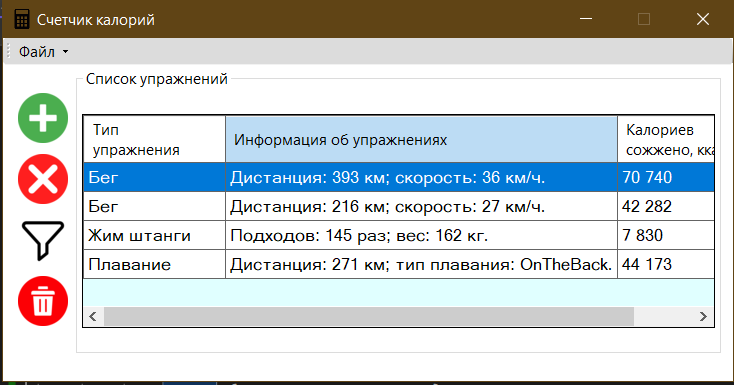


Рисунок 11 – Результат нажатия на значок удаления

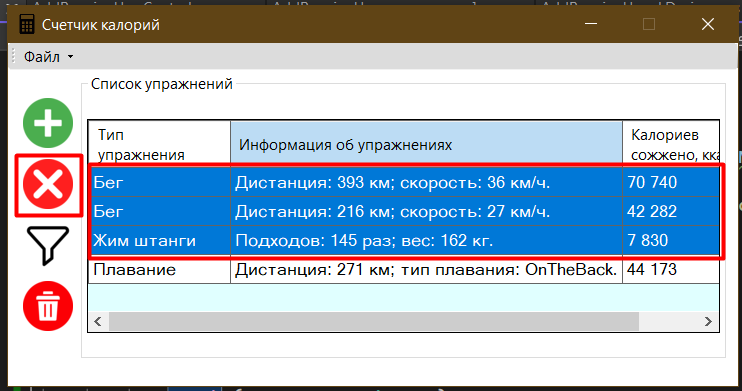


Рисунок 12 – Выбор нескольких элементов для удаления

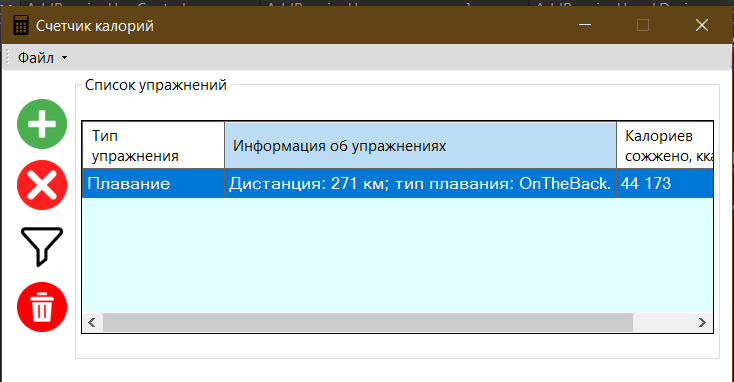


Рисунок 13 – Результат удаления выбранных элементов

**1.5.3 Тестовый случай Фильтра**

Для поиска изданий предусмотрена форма поиска (рисунок 14)

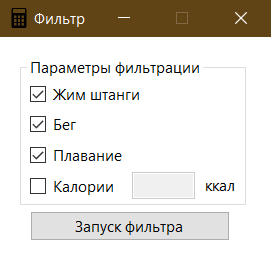


Рисунок 14 – Настройка параметров для поиска упражнений

Пользователь выбирает параметры, по которым требуется найти библиотечное издание, вводит значения этих параметров и нажимает кнопку «Применить» (рисунки 15 и 16).

Для того, чтобы сбросить фильтр, предусмотрена соответствующая кнопка «Сбросить» (рисунок 17).

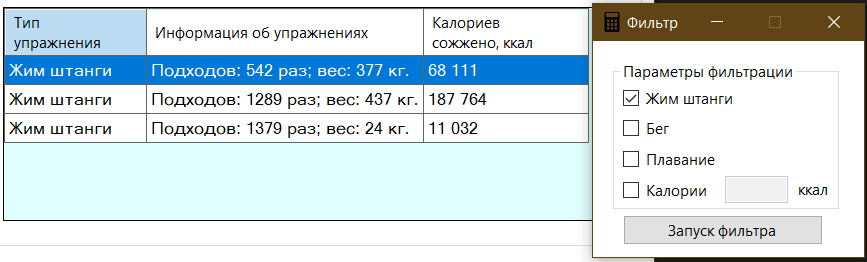


Рисунок 15 – Поиск упражнений «жим штанги»

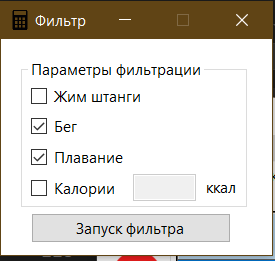
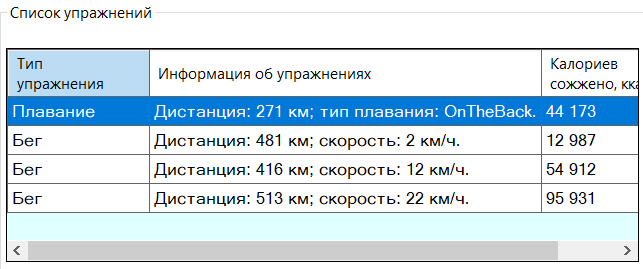
 

Рисунок 16 – Поиск нескольких упражнений

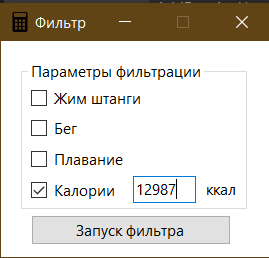
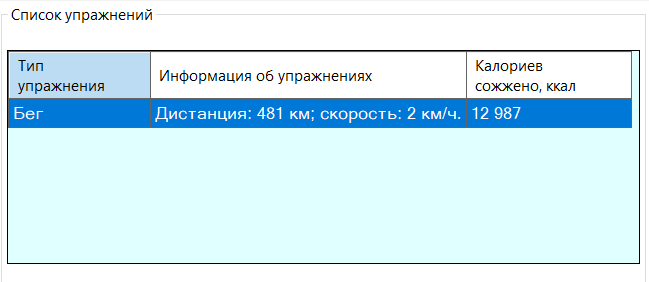
 

Рисунок 17 – Поиск по введенному числу калориев

**1.5.4 Тестовый случай «Сохранить»**

Для сохранения данных в таблице необходимо нажать на кнопку «Сохранить». Откроется системный диалог сохранения файла, где пользователь выбирает директорию и указывает имя файла (рисунок 18).

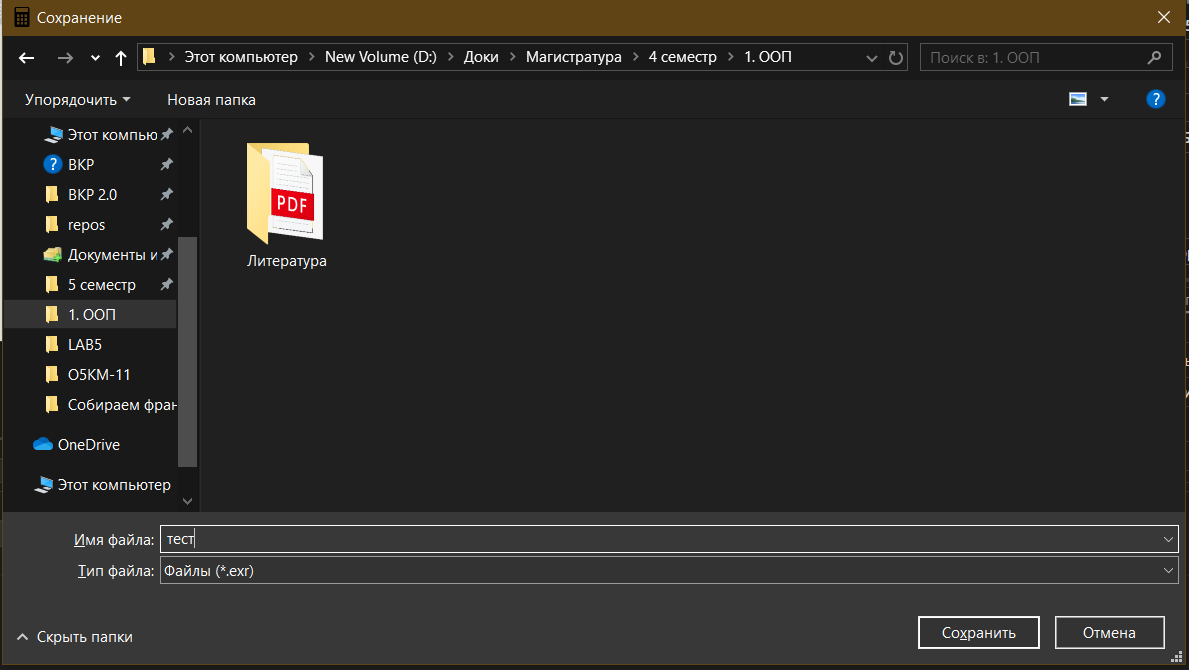


Рисунок 18 – Сохранение файла

**1.5.5 Тестовый случай «Загрузить данные»**

Для загрузки данных в таблицу необходимо нажать на соответствующую кнопку «Загрузить».

Далее откроется системный диалог загрузки файла (рисунок 19).

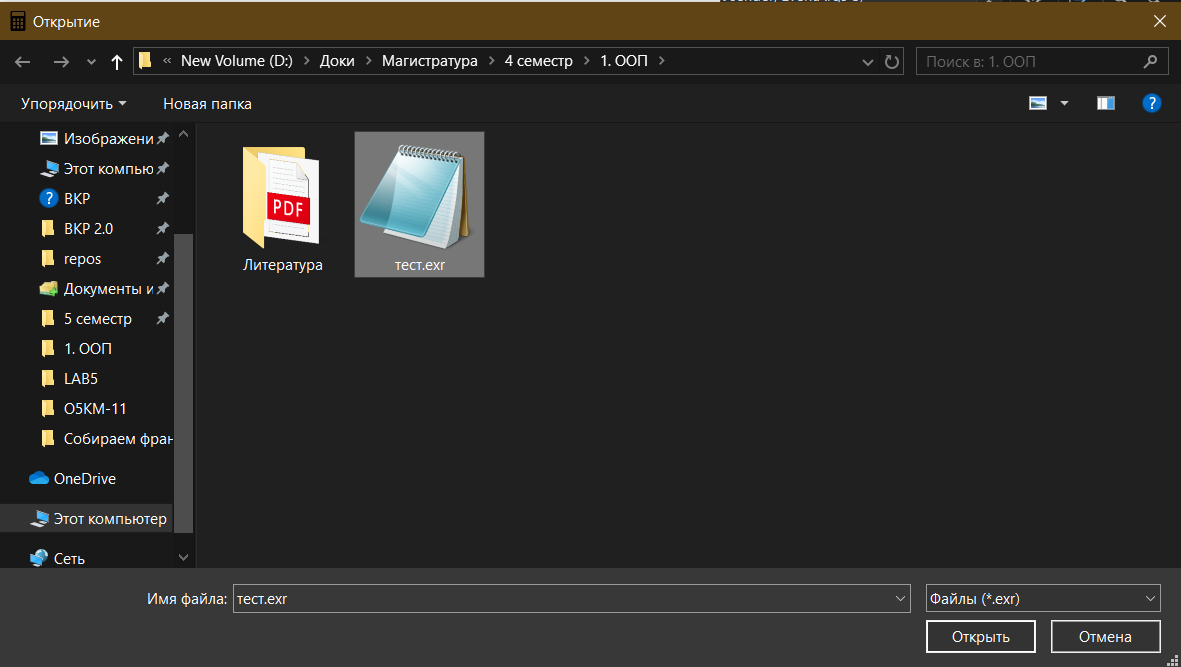


Рисунок 19 – Выбор файла для загрузки

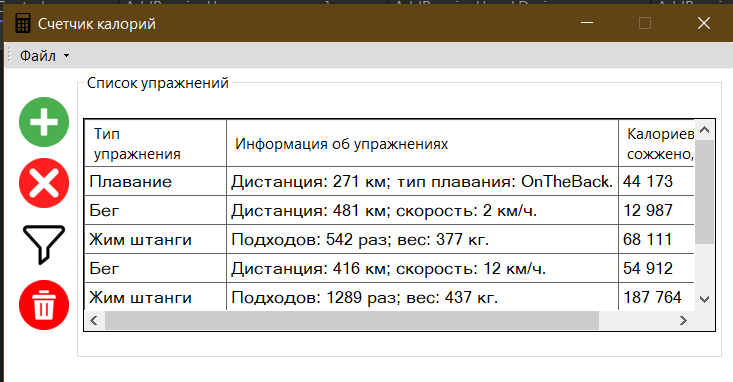


Рисунок 20 – Результат загрузки данных

В случае, если структура данных файла не соответствует установленному формату, появится соответствующее сообщение (рисунок 21).

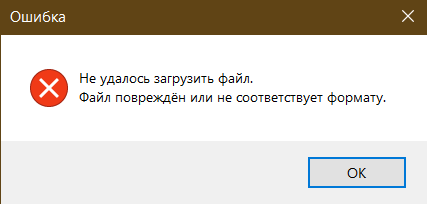


Рисунок 21 – Загрузка повреждённого файла

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Калентьев, А. А. Новые технологии в программировании : учебное пособие / А. А. Калентьев, Д. В. Гарайс, А. Е. Гориянов. – Томск : Эль Контент, 2014. – 176 с. – ISBN 978-5-4332-0185-9.

**ПРИЛОЖЕНИЕ А**

**Техническое задание на создание автоматизированной системы**

Программа для просмотра списка работников

Разработчик: студент гр. О-5КМ11 Братчиков Е.С.

Заказчик: Канд. техн. наук, доцент каф. КСУП ТУСУР Калентьев А. А.

Томск 2023

**1 Общие сведения**

**1.1 Полное наименование системы и её условное обозначение**

Полное наименование: «Программное обеспечение для расчета затраченных калорий при выполнении упражнений».

Условное обозначение: «Система».

**1.2 Наименование предприятий разработчика и заказчика системы**

Заказчик: Канд. техн. наук, доцент каф. КСУП ТУСУР Калентьев А. А.

Разработчик: Студент гр. О-5КМ11 НИ ТПУ Братчиков Е.С.

**1.3 Плановые сроки начала и окончания работы по созданию системы**

Начало работ: 10 апреля 2023 г.

Окончание работ: 10 июня 2023 г.

**2 Назначение и цели создания системы**

**2.1 Назначение системы**

Система предназначена для формирования файла с записями, включающими в себя информацию о упражнениях.

**2.2 Цели создания системы**

1. Унификация процесса формирования упражнений.
2. Уменьшение трудозатрат при расчете числа калорий.

**3 Характеристика объектов автоматизации**

Система рассчитывает затраченные калории при выполнении упражнений: жим штанги, плавание, бег. Каждый из данных типов упражнений характеризуется различным набором информации, необходимую для осуществления расчета. Методики расчета числа затраченных калорий представляют из себя закрытую информацию и не подлежат распространению.

**4 Требования к системе**

Таблица 4.1 – Префиксы мнемонических идентификаторов требований и их расшифровка

|  |  |
| --- | --- |
| **Префикс** | **Тип требования** |
| A | Архитектурное требование |
| C | Требование к аппаратной или программной совместимости |
| F | Функциональное требование |
| U | Требование к пользовательскому интерфейсу |

Таблица 4.2. Требования к системе

|  |  |
| --- | --- |
| **Код** | **Требование** |
| А1 | Система должна быть реализована в виде настольного приложения. |
| С1 | Система должна быть совместима с ОС «Windows» 10 Professional. |
| С2 | Система должна быть совместима с платформой .NET Framework 6.0 |
| С3 | Система должна работать с процессором Intel Pentium 4 или более поздней версии. |
| С4 | Система должна работать с объёмом оперативной памяти – 400 Мб и выше. |
| С5 | Система должна занимать не более 50 Мб пространства на жестком диске. |
| F1 | Система должна обеспечивать ввод исходных данных c клавиатуры с помощью графических форм. |
| F2 | Система должна обеспечивать сохранение и загрузку данных из файла формата XML по запросу пользователя. |
| F3 | Система должна обеспечивать поиск информации по запросу пользователя. |
| F4 | Система должна обеспечивать удаление записей по запросу пользователя. |
| D1 | Система должна обеспечивать проверку корректности введённых данных. |
| U1 | Ввод исходных данных в систему должен производиться с помощью графического интерфейса пользователя. |
| U2 | В интерфейсе должно быть выполнено единообразие элементов для схожих задач. |
| U3 | В интерфейсе результаты ввода должны быть отображены в виде списка. |