|  |
| --- |
| **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-5КМ21 \_\_\_\_\_\_\_\_\_\_ Захаркина В. И. 05 декабря 2024 г.

Отчёт принял доцент, к.т.н. \_\_\_\_\_\_\_\_\_\_ Калентьев А. А. \_\_\_\_\_\_\_ 2024 г.

Томск 2024

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc183976038)

[1. UML диаграмма вариантов использования 4](#_Toc183976039)

[2. UML диаграмма классов 5](#_Toc183976040)

[3. Описание классов, образующих связь типа «общее-частное» 7](#_Toc183976041)

[4. Дерево ветвлений Git 9](#_Toc183976042)

[5. Тестирование программы 10](#_Toc183976043)

[5.1. Тестовый случай «Добавить элемент» 10](#_Toc183976044)

[5.2. Тестовый случай «Удалить элемент» 12](#_Toc183976045)

[5.3. Тестовый случай «Очистить список» 14](#_Toc183976046)

[5.4. Тестовый случай «Найти элемент» 15](#_Toc183976047)

[5.5. Тестовый случай «Сброс фильтра» 19](#_Toc183976048)

[5.6. Тестовый случай «Сохранить данные» 19](#_Toc183976049)

[5.7. Тестовый случай «Загрузить данные» 21](#_Toc183976050)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc183976051)

[ПРИЛОЖЕНИЕ А 25](#_Toc183976052)

**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. 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 отражает взаимодействие между действующими лицами и системой или другим объектом. Действующее лицо представляет собой логическую совокупность ролей, которые пользователи системы выполняют в процессе взаимодействия с ней.

На рисунке 1 представлена диаграмма вариантов использования для разработанного ПО.

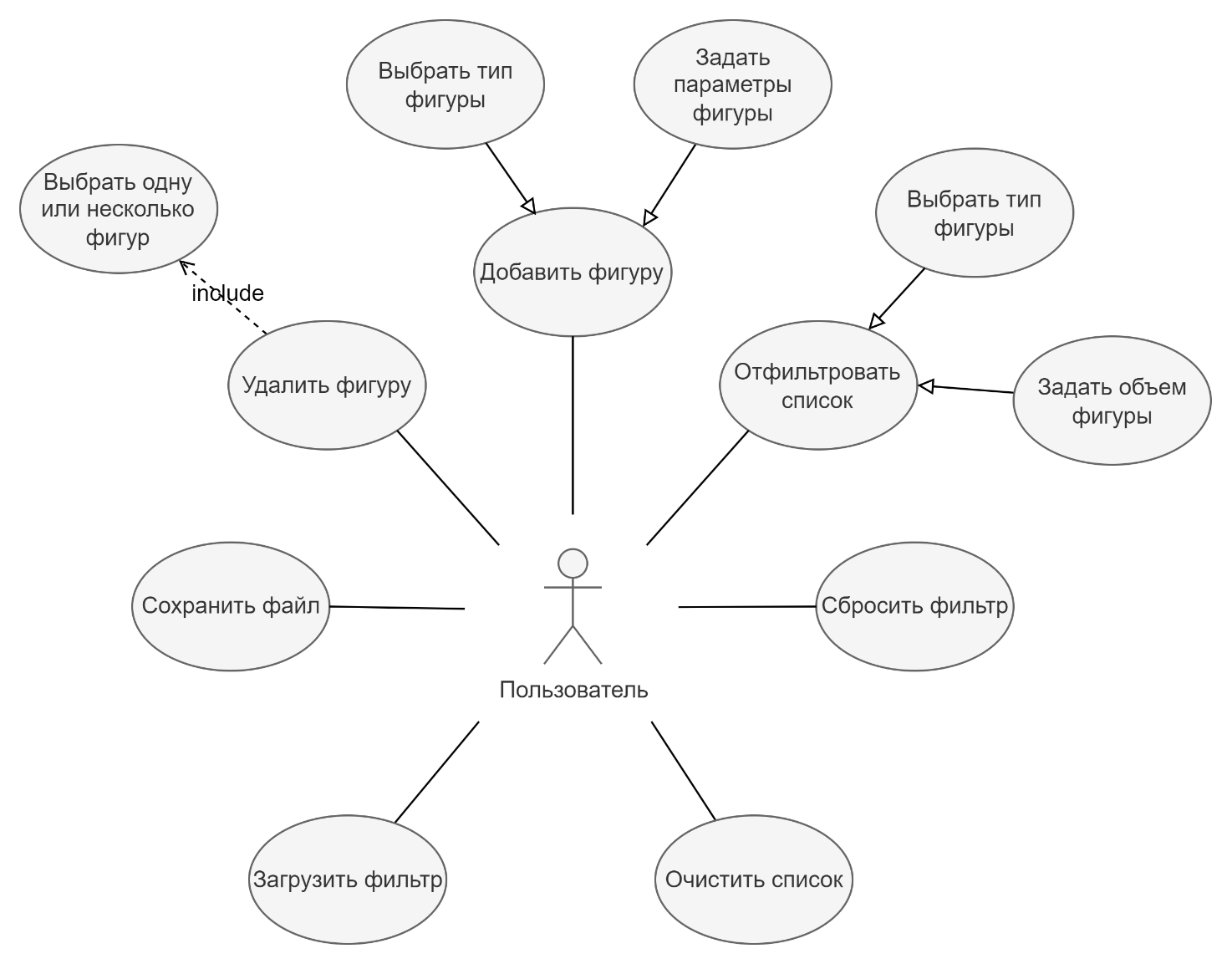


Рисунок 1. Диаграмма вариантов использования

**2. UML диаграмма классов**

Диаграмма классов – это методика моделирования, которая используется практически во всех объектно-ориентированных методах. Эта диаграмма описывает типы объектов в системе и различные виды статических отношений, которые существуют между ними, позволяет наглядно показать, как взаимосвязаны различные элементы системы и как данные передаются и преобразуются внутри нее. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма классов приведена на рисунке 2.

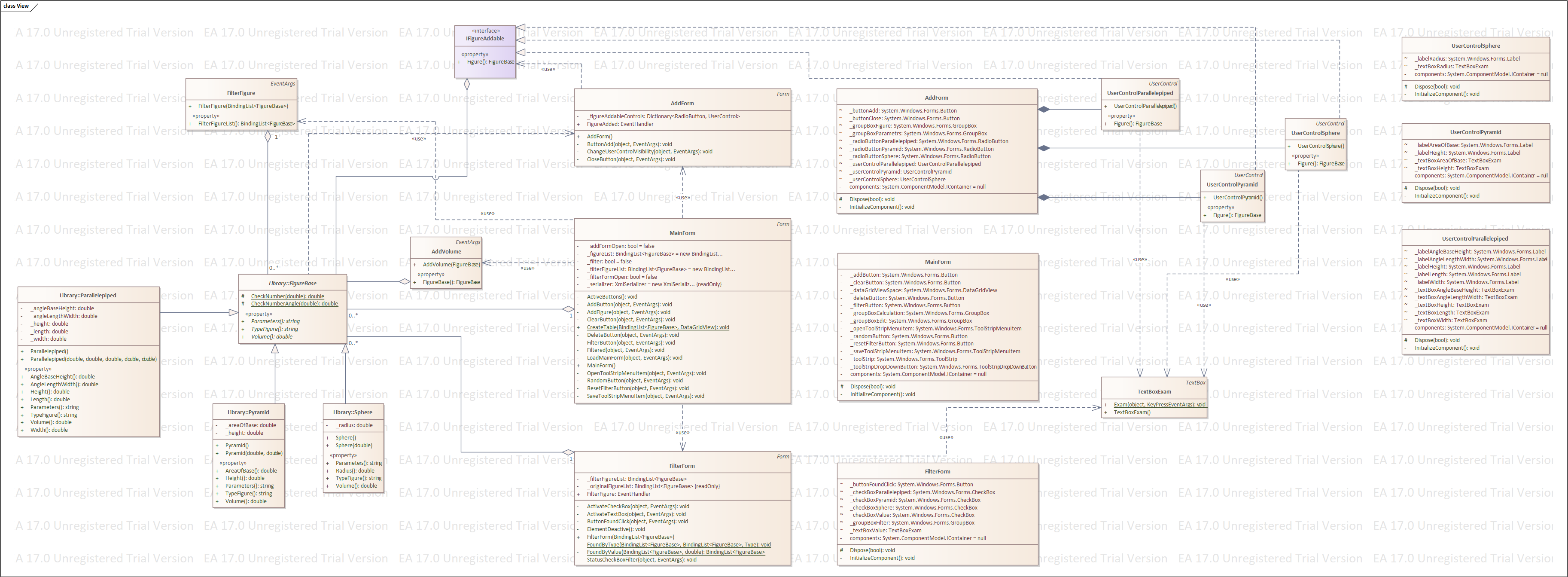


Рисунок 2. UML диаграмма классов

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

В таблице 1 приведено описание интерфейса IFigureAddable для взаимодействия с пользовательскими формами.

Таблица 1. Описание интерфейса *IFigureAddable*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание интерфейса | | |
| Интерфейс *IFigureAddable* – интерфейс для взаимодействия с пользовательскими формами. | | |
| Свойства | | |
| + Figure | FigureBase | Фигура.  Объект абстрактного класса FigureBase. |

В таблицах 2 – 4 приведены описания классов UserControlSphere, UserControlParallelepiped и UserControlPyramid.

Таблица 2. Описание класса *UserControlSphere*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание класса | | |
| Класс *UserControlSphere* – класс формы *Sphere* | | |
| Свойства | | |
| + Figure | FigureBase | Чтение параметров фигуры Sphere. |

Таблица 3. Описание класса *UserControlParallelepiped*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *UserControlParallelepiped* – класс формы *Parallelepiped* | | |
| Свойства | | |
| + Figure | FigureBase | Чтение параметров фигуры Parallelepiped. |

Таблица 4. Описание класса *UserControlPyramid*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание класса | | |
| Класс *UserControlPyramid* – класс формы *Pyramid* | | |
| Свойства | | |
| + Figure | FigureBase | Чтение параметров фигуры Pyramid. |

В таблице 5 приведено описание абстрактного класса FigureBase с его полями, свойствами и методами.

Таблица 5. Описание класса *FigureBase*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание класса | | |
| Класс *FigureBase* – абстрактный базовый класс для фигур. | | |
| Свойства | | |
| + TypeFigure | string | Вид фигуры.  Абстрактное свойство, переопределяется в производных классах. |
| + Parameters | string | Параметры фигуры.  Абстрактное свойство, переопределяется в производных классах. |
| + Volume | double | Объем фигуры.  Абстрактное свойство, переопределяется в производных классах. |
| Методы | | |
| + CheckNumber | double | Проверка ввода значения радиуса, длины, ширины, высоты. |
| + CheckNumberAngle | double | Проверка ввода значения угла между длиной и шириной, угла между основанием и высотой. |

В таблицах 6 – 8 приведены описания классов *Sphere*, *Parallelepiped* и *Pyramid*, которые наследуются от *FigureBase*.

Таблица 6. Описание класса *Sphere*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание класса | | |
| Класс *Sphere* – шар. | | |
| Поля | | |
| – \_radius | double | Радиус шара. |
| Свойства | | |
| + TypeFigure | string | Вид фигуры. |
| + Parameters | string | Параметры фигуры. |
| + Radius | double | Радиус шара. |
| + Volume | double | Вычисление объема шара. |

Таблица 7. Описание класса *Parallelepiped*

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Описание класса | | |
| Класс *Parallelepiped* – параллелепипед. | | |
| Поля | | |
| – \_angleLengthHeight | double | Угол между основанием и высотой параллелепипеда. |
| – \_angleLengthWidth | double | Угол между длиной и шириной параллелепипеда. |
| – \_height | double | Высота параллелепипеда. |
| – \_length | double | Длина параллелепипеда. |
| – \_width | double | Ширина параллелепипеда. |
| Свойства | | |
| + TypeFigure | string | Вид фигуры. |
| + Parameters | string | Параметры фигуры. |
| + AngleLengthHeight | double | Угол между основанием и высотой параллелепипеда. |
| + AngleLengthWidth | double | Угол между длиной и шириной параллелепипеда. |
| + Height | double | Высота параллелепипеда. |
| + Length | double | Длина параллелепипеда. |
| + Width | double | Ширина параллелепипеда. |
| + Volume | double | Вычисление объема параллелепипеда. |

Таблица 8. Описание класса *Pyramid*

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| Описание класса | | |
| Класс *Pyramid* – пирамида. | | |
| Поля | | |
| – \_areaOfBase | double | Площадь основания пирамиды. |
| – \_height | double | Высота пирамиды. |
| Свойства | | |
| + AreaOfBase | double | Площадь основания пирамиды. |
| + Height | double | Высота пирамиды. |
| + TypeFigure | string | Вид фигуры. |
| + Parameters | string | Параметры фигуры. |
| + Volume | double | Вычисление объема пирамиды. |

**4. Дерево ветвлений Git**

Git – распределённая система управления версиями файлов, спроектирована как набор утилит командной строки, специально разработанных с учётом их использования в скриптах [2].

Данная система позволяет реализовать эффективную совместную работу над проектом и управление версиями кода.

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

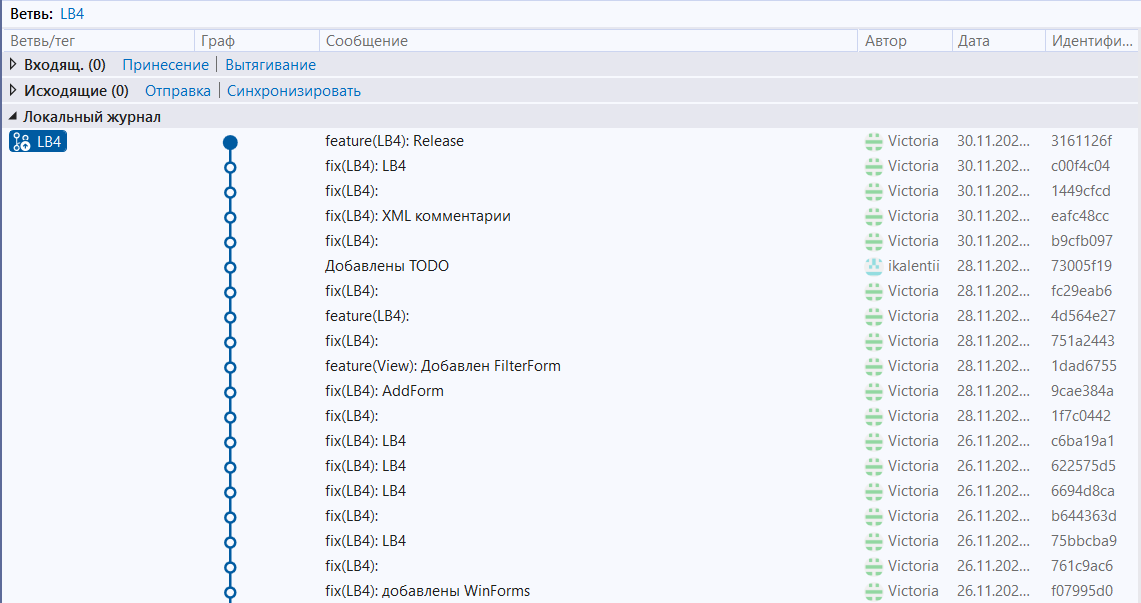


Рисунок 3. Дерево ветвлений Git

**5. Тестирование программы**

Далее приводится процесс функционального тестирования программы.

На рисунке 4 представлен графический интерфейс пользователя.

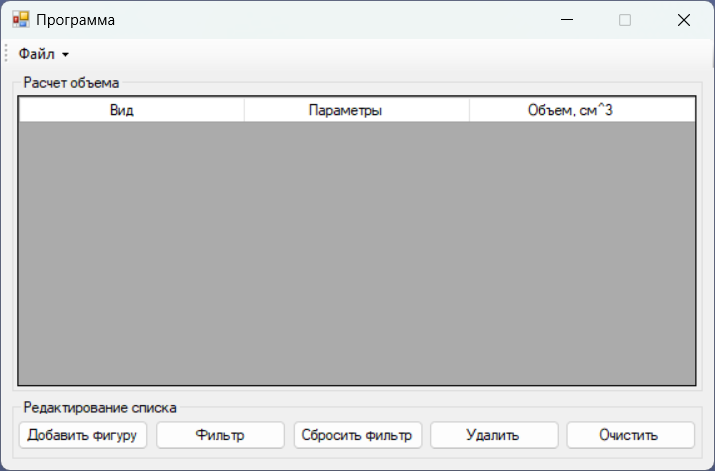


Рисунок 4. Графический интерфейс пользователя

**5.1. Тестовый случай «Добавить элемент»**

Кнопка «Добавить фигуру» позволяет вызвать соответствующую форму для добавления нового элемента, которая изображена на рисунке 5.

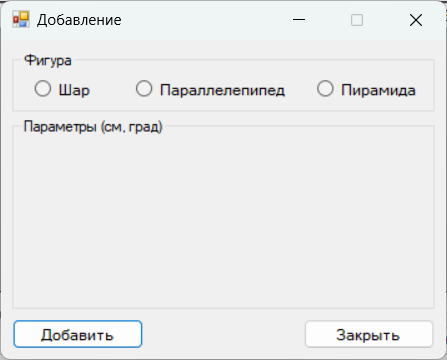


Рисунок 5. Форма для добавления элемента

При выборе необходимой фигуры можно ввести исходные данные (параметры фигуры). После ввода данных необходимо нажать кнопку «Добавить». После этого добавленный элемент появится в таблице главной формы (рисунки 6 и 7).

Программа реализована таким образом, что недопускает вводить пользователю символы, отличных от цифр. Также программа не даст ввести число с несколькими запятыми.

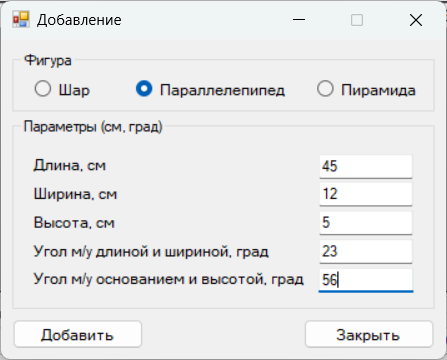


Рисунок 6. Заполнение полей

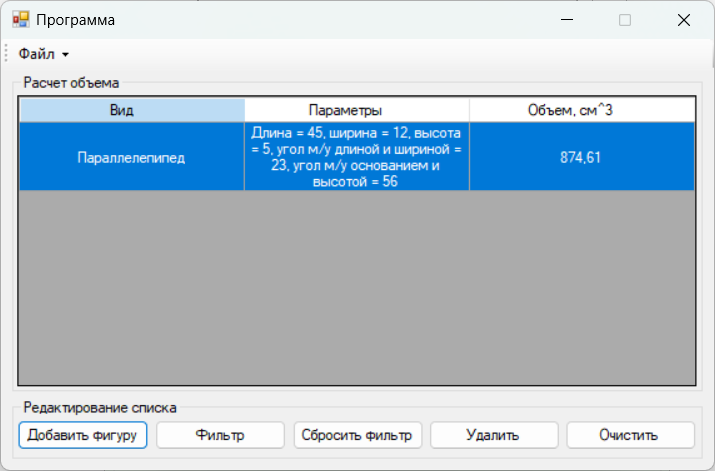


Рисунок 7. Успешное добавление нового элемента

В случае, если пользователь не заполнил требуемые поля, то программа выдает предупреждение «Введите данные.» (рисунки 8 и 9).

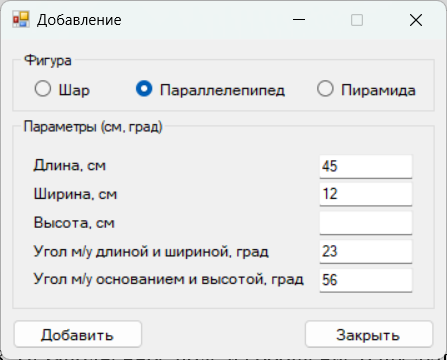


Рисунок 8. Незаполненное поле

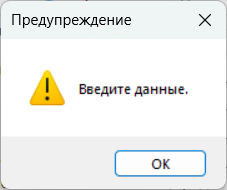


Рисунок 9. Предупреждение

В случае ввода в поля «Угол м/у длиной и шириной, град» и «Угол м/у основанием и высотой, град» значений меньше 0 или больше 180, программа выдает предупреждение «Угол должен быть от 0 до 180 град.» (рисуноки 10 и 11).

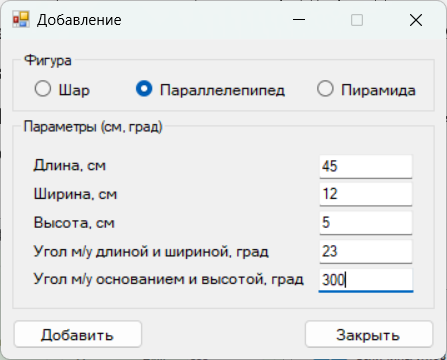


Рисунок 10. Некорректный ввод данных

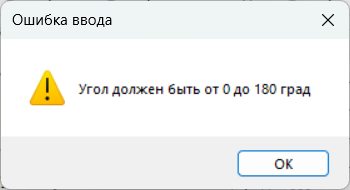


Рисунок 11. Ошибка ввода

**5.2. Тестовый случай «Удалить элемент»**

Кнопка «Удалить» позволяет пользователю удалить один или несколько элементов при выборе их в таблице (рисунки 12 – 15).

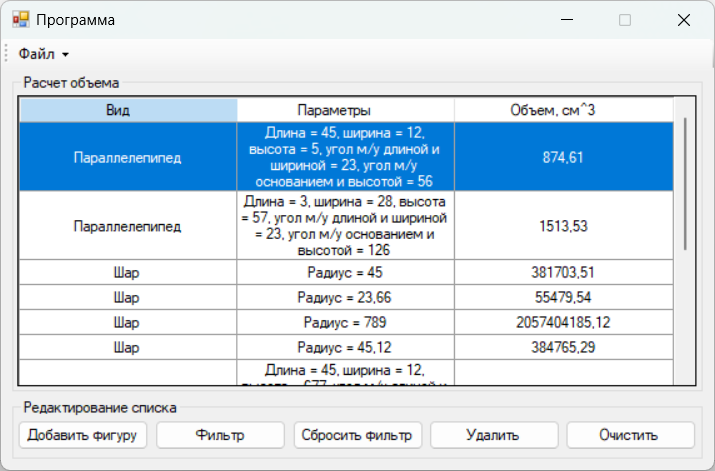


Рисунок 12. Выбор элемента в таблице

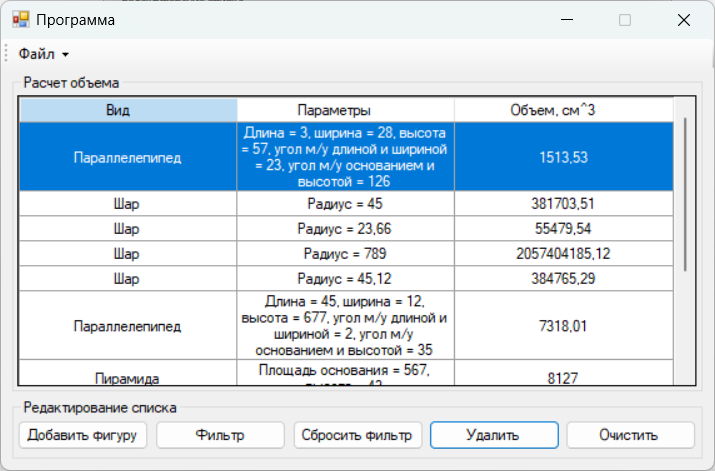


Рисунок 13. Результат нажатия кнопки «Удалить»

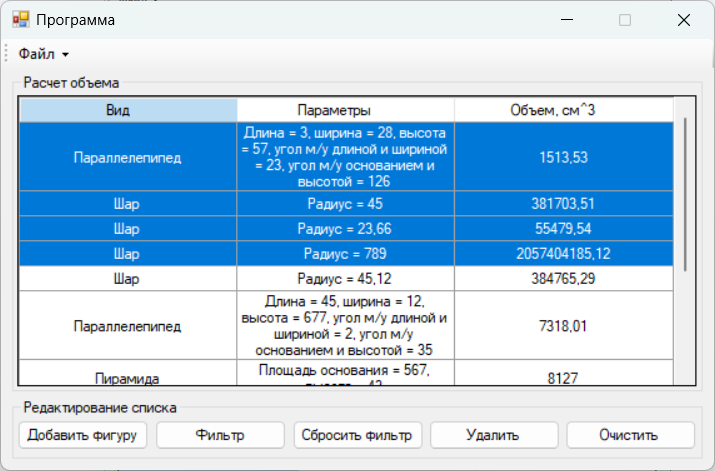


Рисунок 14. Выбор нескольких элементов для удаления

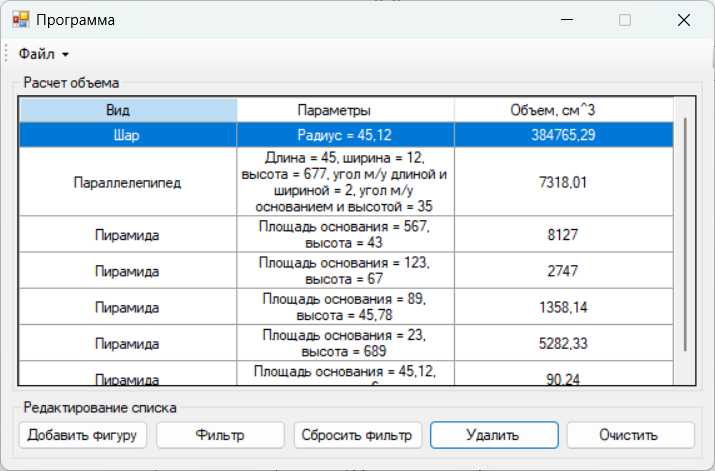


Рисунок 15. Результат удаления выбранных элементов

**5.3. Тестовый случай «Очистить список»**

Кнопка «Очистить» позволяет пользователю полностью очистить список (рисунок 16 и 17).

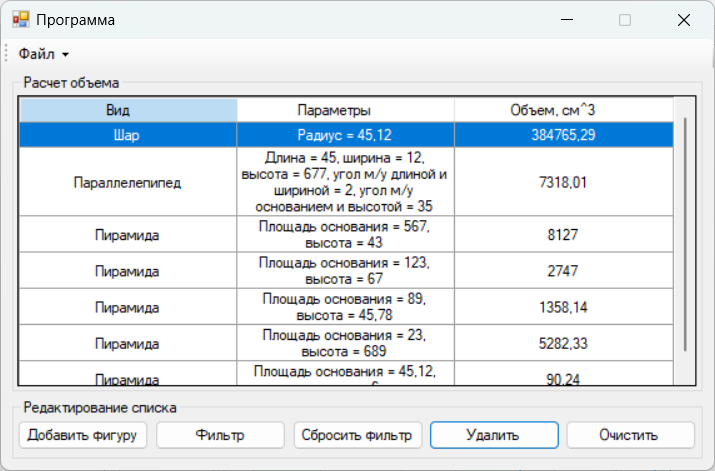


Рисунок 16. Исходный список до нажатия кнопки «Очистить»

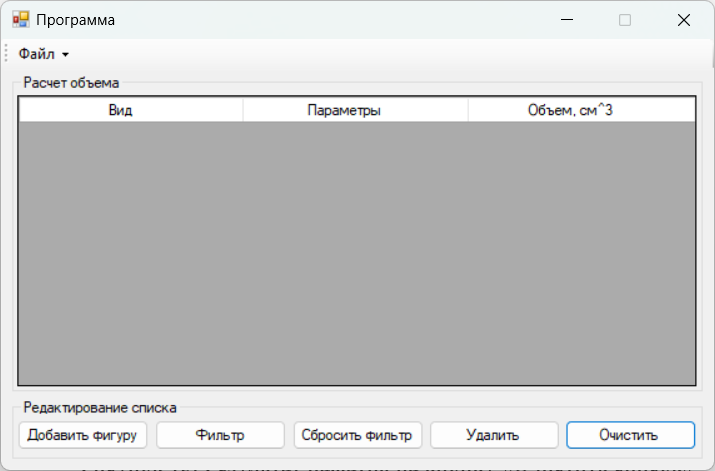


Рисунок 17. Результат нажатия на кнопку «Очистить»

**5.4 Тестовый случай «Фильтрация»**

Кнопка «Фильтр» позволяет пользователю произвести фильтрацию фигур по их виду и значению объема. При нажатии на кнопку откроется соответствующая форма для поиска (рисунок 18).

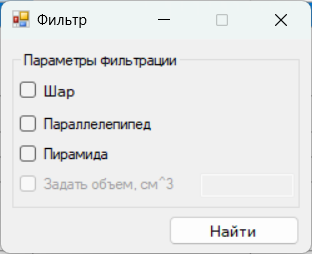


Рисунок 18. Вызов формы для фильтрации списка фигур

Ниже приведены примеры фильтрации фигур:

* «Шар»

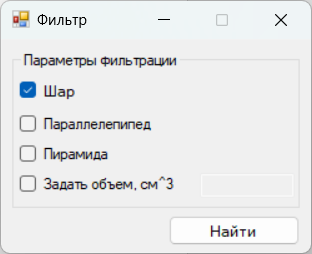


Рисунок 19. Поиск фигур «Шар»

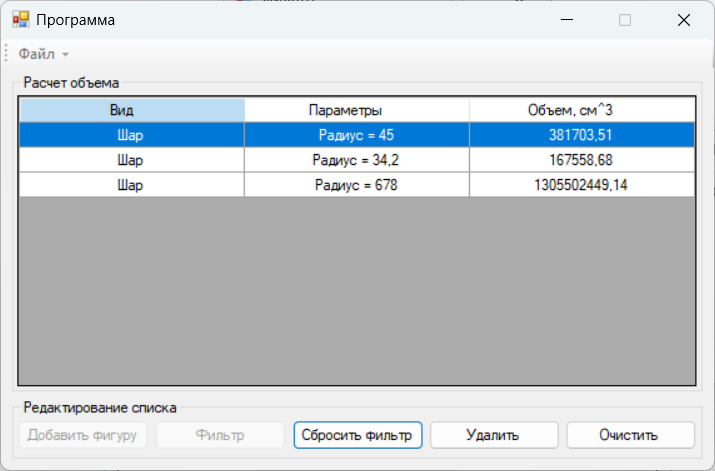


Рисунок 20. Результат поиска фигур «Шар»

* «Параллелепипед»

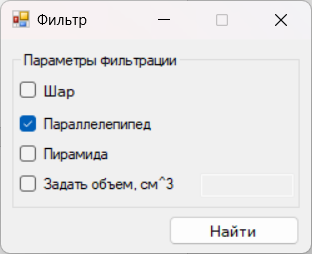


Рисунок 21. Поиск фигур «Параллелепипед»

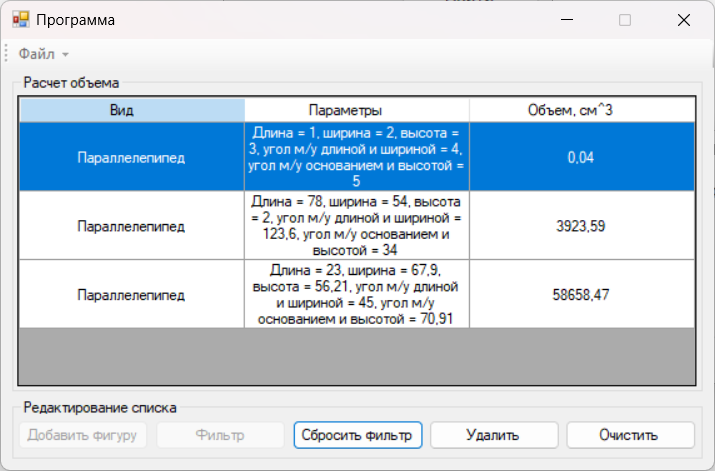


Рисунок 22. Результат поиска фигур «Параллелепипед»

* «Пирамида»

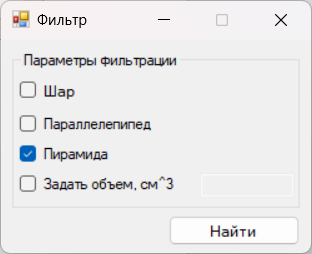


Рисунок 23. Поиск фигур «Пирамида»

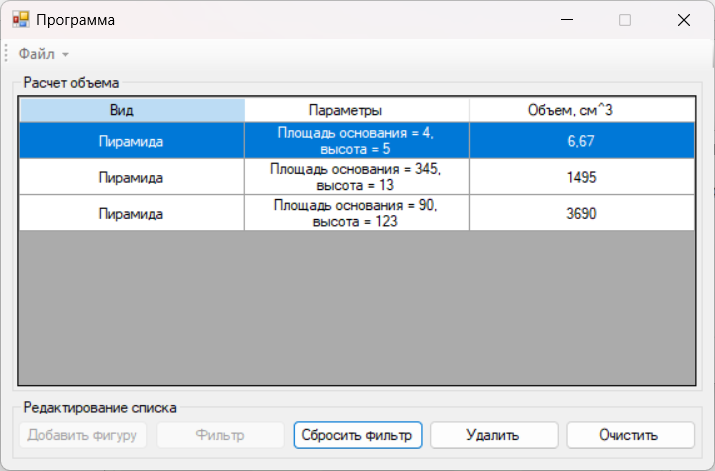


Рисунок 24. Результат поиска фигур «Пирамида»

* «Задать объем, см^3»

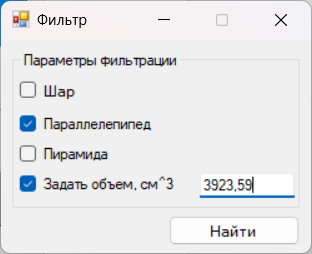


Рисунок 25. Поиск фигуры по параметру «Задать объем, см^3»

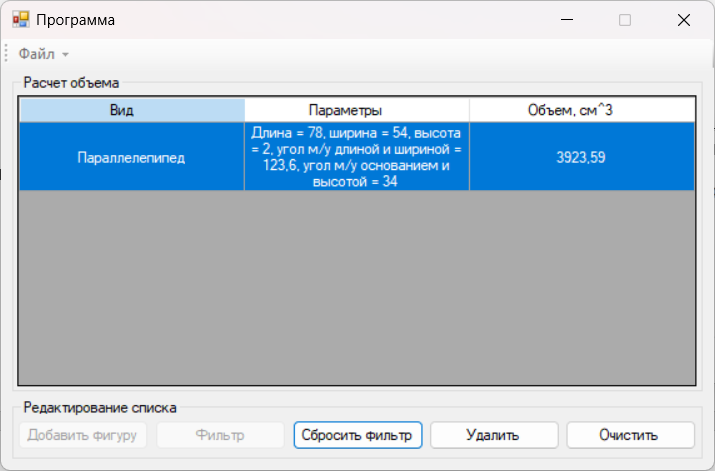


Рисунок 26. Результат поиска по параметру «Задать объем, см^3»

Если пользователь задал некорректные условия для фильтрации, то Система среагирует на это в виде предупреждения (рисунки 27 и 28).

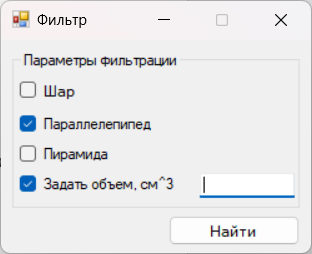


Рисунок 27. Нажатие кнопки «Найти» при незаполненном поле

«Задать объем, см^3»

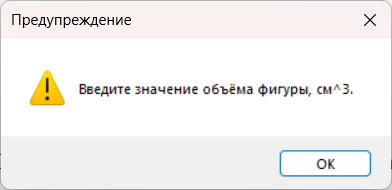


Рисунок 28. Предупреждение пользователя при незаполненном поле

«Задать объем, см^3»

**5.5. Тестовый случай «Сброс фильтра»**

На рисунках 29 и 30 представлена работа функции сброса фильтра.

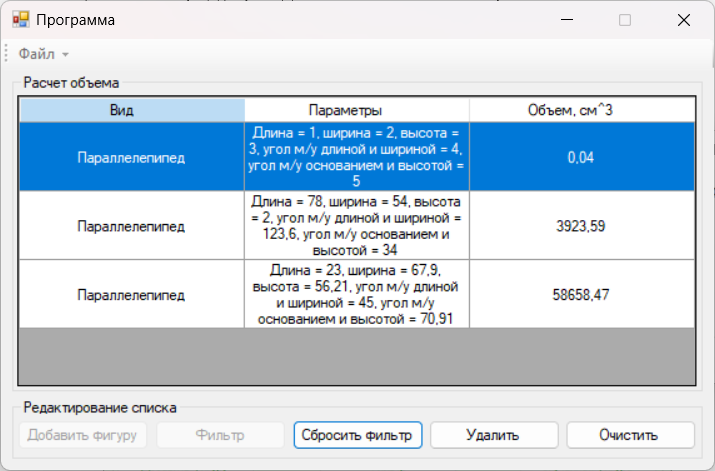


Рисунок 29. Отфильтрованный список до нажатия на кнопку «Сбросить фильтр»

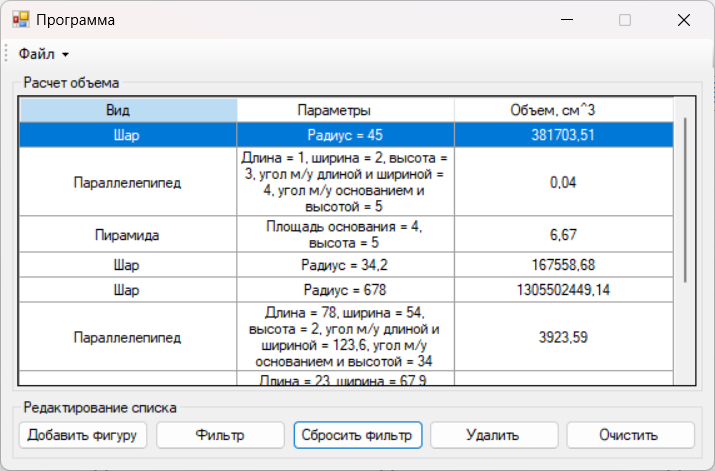


Рисунок 30. Отображение исходного списка после сброса фильтра

**5.6. Тестовый случай «Сохранить данные»**

Для сохранения данных в таблице необходимо нажать на панели «Файл» 🡪 «Сохранить» (рисунок 31). Откроется системный диалог, где пользователь выбирает директорию и указывает имя файла (рисунок 32).

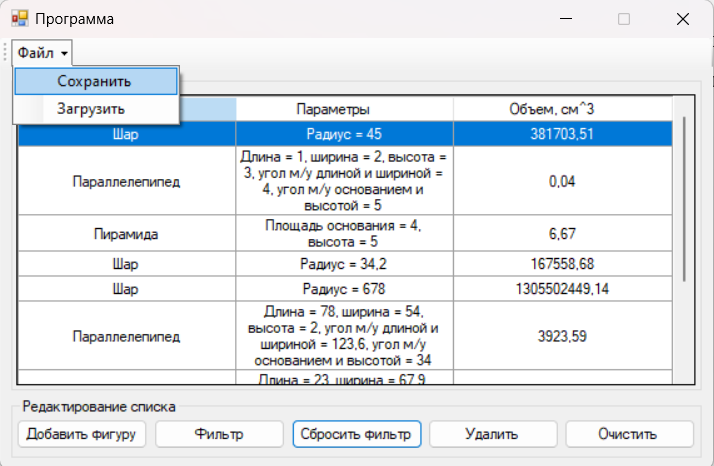


Рисунок 31. Сохранение данных

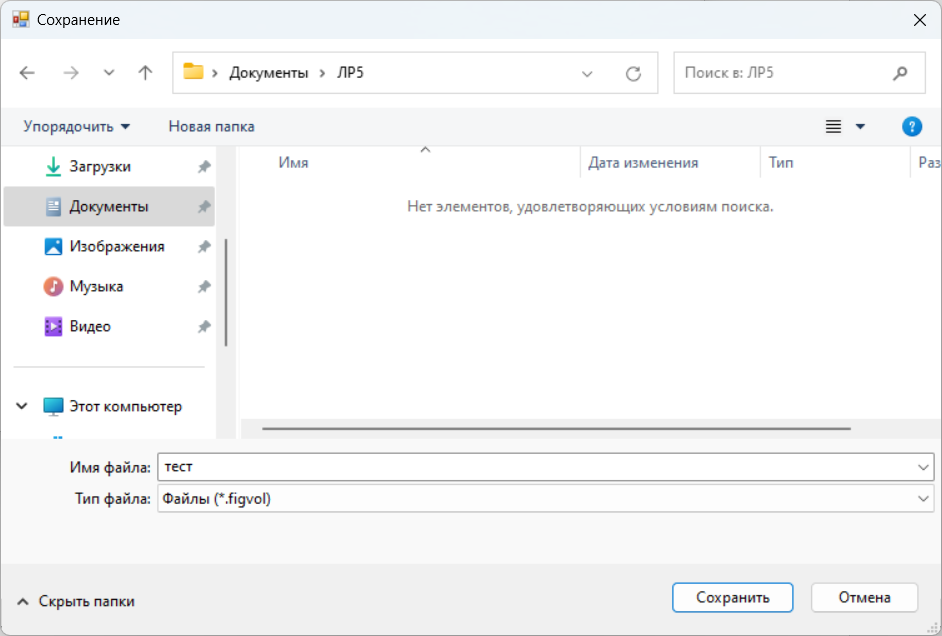


Рисунок 32. Сохранение файла

После сохранения данных в файл появится соответствующее сообщение (рисунок 33) На рисунке 34 представлен результат процедуры.

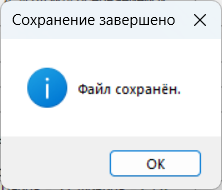


Рисунок 32. Сообщение о сохранении файла

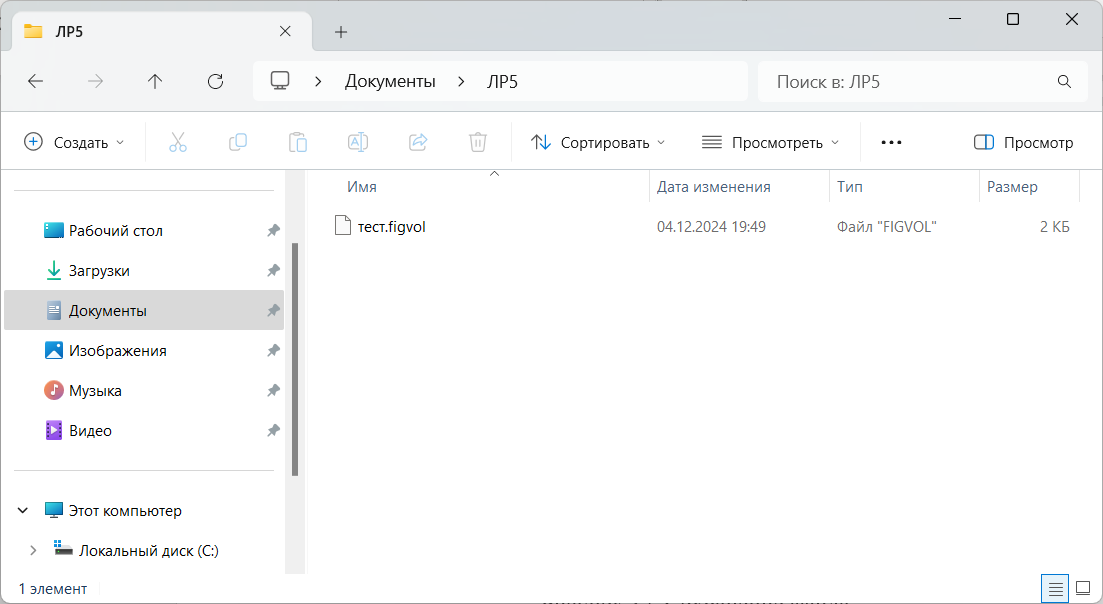


Рисунок 34. Результат сохранения файла

В случае, если таблица пуста, сохранение не производится (рисунок 35).

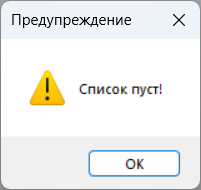


Рисунок 35. Предупреждение при сохранении пустой таблицы

**5.7. Тестовый случай «Загрузить данные»**

Для загрузки данных в таблицу необходимо нажать на соответствующую кнопку «Файл» 🡪 «Загрузить» (рисунок 36).

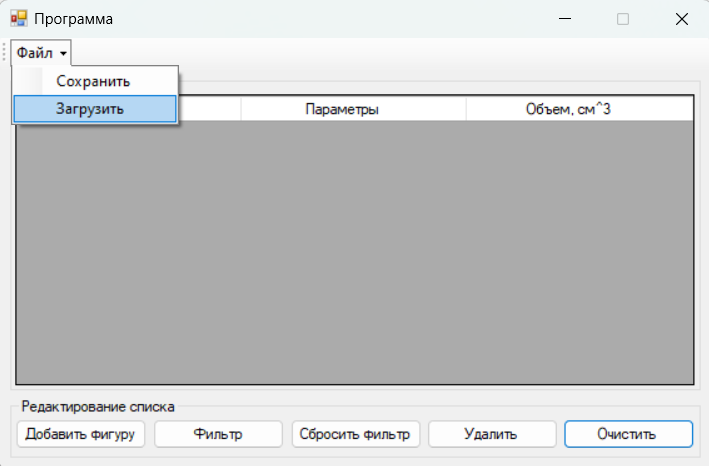


Рисунок 36. Загрузить данные в таблицу

Далее откроется системный диалог (рисунок 37).

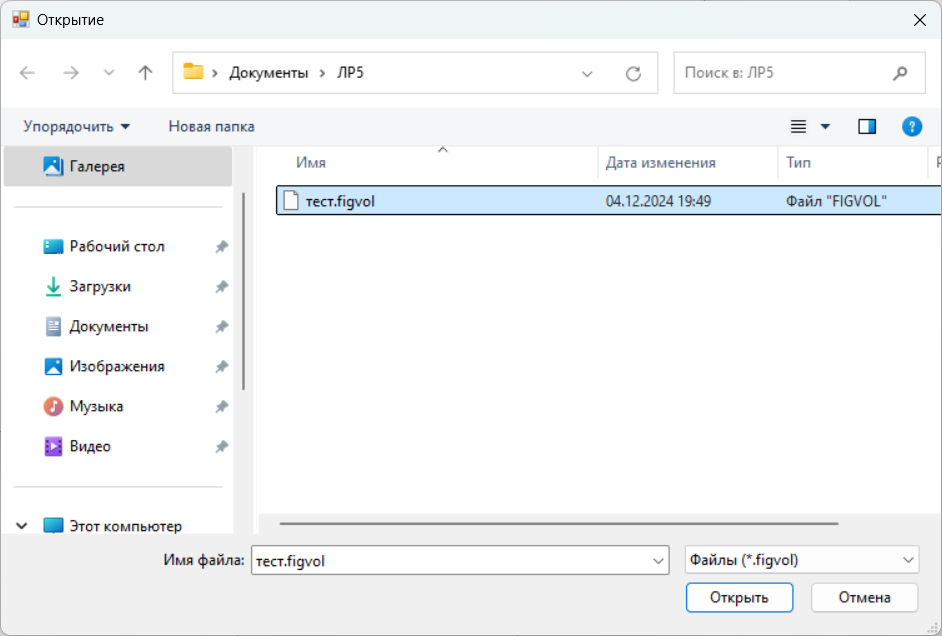


Рисунок 37. Выбор файла для загрузки

На рисунках 38 и 39 представлен результат процедуры и соответствующее ему сообщение.

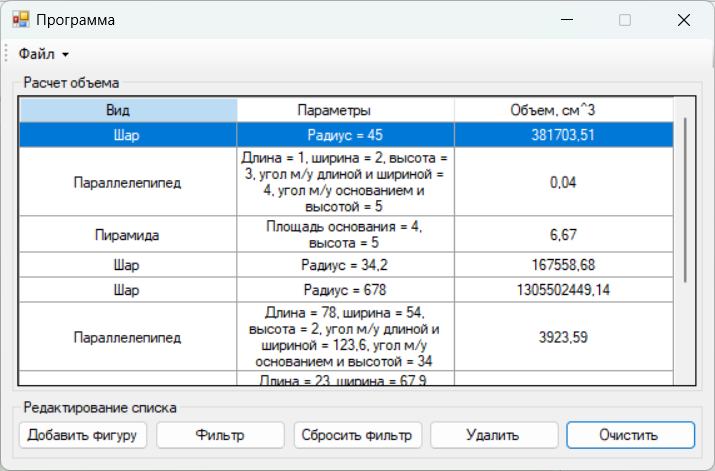


Рисунок 38. Результат загрузки файла

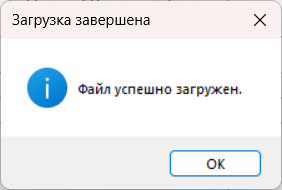


Рисунок 39. Сообщение после загрузки файла

На рисунке 40 представлено сообщение, которое предупреждает пользователя о том, что структура загружаемого файла не распознана, или в файле присутствуют некорректные значения параметров, или значения отсутствуют.

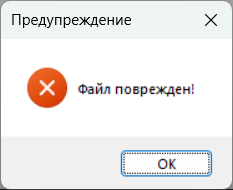


Рисунок 40. Предупреждение

**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.

2. Git [Электронный ресурс]. — URL : http://git-scm.com/ (Дата обращения: 16.10.2024)

**ПРИЛОЖЕНИЕ А**

**Техническое задание на создание автоматизированной системы**

Программа расчёта объема фигур

Разработчик: студент гр. О-5КМ21 НИ ТПУ Захаркина В. И.

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

Томск 2024

**1. Общие сведения**

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

Полное наименование: «Программа расчёта объема фигуры».

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-5КМ21 НИ ТПУ Захаркина В. И.

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

Начало работ: 21 мая 2024 г.

Окончание работ: 05 декабря 2024 г.

**2. Назначение и цели создания системы**

**2.1. Назначение системы**

Система предназначена для расчёта объема фигур: шар, параллелепипед, пирамида.

**2.2. Цели создания системы**

Цель создания Системы заключается в автоматизации и упрощении процесса вычисления объема различных фигур (шар, параллелепипед, пирамида), а также заключается в повышении эффективности работы пользователя с геометрическими данными и упрощении процесса выполнения вычислений.

**3. Характеристика объектов автоматизации**

Вычисление объёмов фигур – это процесс нахождения пространства, которое занимает трёхмерный объект. Полученный результат расчета (объем фигуры) используется для многих практических задач, таких как строительство и архитектура, инженерия и машиностроение, физика и химия, а также кулинария и пищевая промышленность

Поскольку такие расчёты необходимы везде, где требуется точное понимание пространства, которое занимает или может занять объект, и выполняются многочисленное количество раз, представляется целесообразным автоматизировать этот процесс.

**4. Требования к системе**

Таблица 4.1. Префиксы мнемонических идентификаторов требований и их расшифровка

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

**4.1. Требования к архитектуре**

**А01.** Система должна быть реализована в виде настольного приложения.

**4.2. Требования к структуре данных**

**D01.** Данные о параметрах фигуры должны храниться в формате \*.figvol.

Структура XML-документа:

<?xml version="1.0"?>

<ArrayOfFigureBase> <!--Корневой объект-->

<FigureBase> <!--Элемент, определяющий способ расчета объема фигуры-->

<!--Параметры, соответствующие типу фигуры-->

</FigureBase>

</ArrayOfFigureBase>

* Шар

<FigureBase xsi:type="Ball">

<Radius> <!--Радиус, double--> </Radius>

</FigureBase>

* Параллелепипед

<FigureBase xsi:type="Parallelepiped">

<Length> <!--Длина, double--> </Length>

<Width> <!--Ширина, double--> </Width>

<Height> <!--Высота, double--> </Height>

<AngleLengthWidth> <!--Угол α, double--> </AngleLengthWidth>

<AngleLengthHeight> <!--Угол β, double--> </AngleLengthHeight>

</FigureBase>

* Пирамида

<FigureBase xsi:type="Pyramid">

<AreaOfBase> <!--Площадь основания, double--> </AreaOfBase>

<Height> <!--Высота, double--> </Height>

</FigureBase>

**D02.** Система должна принимать входные данные и формировать выходные данные в следующих единицах измерения: сантиметр, градус.

**4.3. Функциональные требования**

**F01.** Система должна рассчитывать фигур следующих типов:

* шар;
* параллелепипед;
* пирамида.

**F01.01.** Объем шара должен определяться по выражению

где *r* – радиус шара (*r* > 0), м;

**F01.02.** Объем параллелепипеда должен определяться по выражению

где *l* – длина параллелепипеда (*l* > 0), м;

*h* – высота параллелепипеда (*h* > 0), м;

– ширина параллелепипеда (*w* > 0), м;

– угол между длиной и шириной параллелепипеда (*α* > 0°; *α* < 180°), град.;

– угол между основанием и высотой параллелепипеда (*β* > 0°; *β* < 180°), град.

**F01.03.** Объем пирамиды должен определяться по выражению

где *a* – площадь основания пирамиды, м^2;

*h* – высота пирамиды, м.

**F02.** В системе должен быть реализован список элементов фигур.

**F02.01.** Каждый элемент должен иметь следующие параметры:

* тип фигуры;
* объем.

**F03.** В системе должна присутствовать функция добавления элементов в список.

**F04.** В системе должна присутствовать функция удаления элементов из списка.

**F05.** В системе должна присутствовать функция очистки списка.

**F06.** В системе должна присутствовать функция поиска элементов по параметрам, указанным в **F02.01**.

**F07.** В системе должна присутствовать функция сохранения списка элементов в файл (**D01**).

**F08.** В системе должна присутствовать функция загрузки списка элементов из файла (**D01**).

**4.4. Требования к пользовательскому интерфейсу**

**U01.** Система должна иметь графический интерфейс пользователя.

**U02.** Данные должны быть представлены в табличном виде.

**U03.** В системе должна быть реализована система обработки ошибок.

**4.5. Требования к программному обеспечению**

**C01.** Система должна работать на операционной системе 64-бит Windows 11 версии 22H2. Работоспособность на других выпусках и версиях не гарантируется.

**C02.** На рабочей станции должен быть установлен .NET Framework версии 4.7.2.

**4.6. Требования к аппаратному обеспечению**

**C03.** Частота процессора не менее 1000 МГц.

**C04.** Объём оперативной памяти не менее 2 ГБ для 64-разрядных систем и 1 ГБ для 32-разрядных систем.

**C05.** Требуемый объём занимаемой памяти 50 МБ.