

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Кубанский государственный университет»**

**(ФГБОУ ВО «КубГУ»)**

**Институт среднего профессионального образования**

**(ИНСПО)**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**«МДК 03.01 Технология разработки**

**программного обеспечения» на тему:**

**Разработка программы расчета энергоснабжения дома**

Выполнила студентка гр. ПКС-1

Соболева Полина Арсеньевна

Руководитель преподаватель

Трубников Ю.Ю.

Краснодар, 2022

СОДЕРЖАНИЕ

**[ВВЕДЕНИЕ](#_Toc119259791)** [3](#_Toc119259791)

**[1.](#_Toc119259792)****[Анализ задания и выбор технологии, языка и среды разработки](#_Toc119259792)** [4](#_Toc119259792)

**[2.](#_Toc119259793)****[Определение структуры программного продукта](#_Toc119259793)** [9](#_Toc119259793)

**[2.1.](#_Toc119259794)****[Анализ процесса обработки информации и выбор структур данных для ее хранения](#_Toc119259794)** [9](#_Toc119259794)

**[2.2.](#_Toc119259795)****[Выбор методов решения задачи и разработка основных алгоритмов предметной области](#_Toc119259795)** [9](#_Toc119259795)

[2.3 Построение структурной схемы программного продукта 13](#_Toc119259796)

**[3.](#_Toc119259797)****[Разработка интерфейса пользователя](#_Toc119259797)** [14](#_Toc119259797)

**[3.1 Построение диаграммы вариантов использования](#_Toc119259798)** [14](#_Toc119259798)

**[3.2.](#_Toc119259799)****[Разработка форм ввода-вывода информации](#_Toc119259799)** [16](#_Toc119259799)

**[4.](#_Toc119259800)****[Разработка диаграммы классов программы](#_Toc119259800)** [19](#_Toc119259800)

**[5.](#_Toc119259801)****[Разработка диаграммы последовательности](#_Toc119259801)** [20](#_Toc119259801)

# **ВВЕДЕНИЕ**

Развитие человеческой цивилизации на протяжении всей её истории сопровождало непрерывное развитие методов и технологий обработки и использования материалов. Одним из важнейших технологических прорывов для человечества стало изобретение термической обработки пищи, воды, что позволило значительно повысить качество пищи и её усваиваемость.

На пути совершенствования методов термообработки и искусственного обогрева помещений, человечество сменило множество материалов: от брёвен и кизяка, до угля, и, наконец, газа.

Современную жизнь невозможно представить без использования газообразного топлива. Использование природного газа в качестве топлива для отопления и приготовления пищи позволяет не только обеспечить комфортные условия для жизни человека, но и значительно сократить затраты, а также уменьшить выбросы вредных веществ в атмосферу.

Однако массовые потребители данного вида топлива ввиду его дешевизны, могут потреблять сильно больше пропускной способности газового счётчика, таким образом приводя к его неисправности, и в дальнейшем переплачивая на его переустановке.

Чтобы не допустить износа газового оборудования и его дальнейшей замены, верным исходом будет провести подсчёты средних и пиковых расходов топлива. В том числе, это поможет сэкономить не только на амортизации оборудования, но и на непосредственном потреблении газа.

Для массового потребителя газа подобные вычисления могут показаться сложными, т.к. в них достаточно просто допустить ошибку. Поэтому более удобным и надёжным решением в данном вопросе будет использование автоматических вычислений - калькулятор расчётов энергопотребления дома.

# **Анализ задания и выбор технологии, языка и среды разработки**

Цель работы состоит разработке приложения "Калькулятор для расчета энергоснабжения дома", которое могло бы выполнять арифметические действия с выбранными и вводимыми пользователем данными.

К полученным результатам относится разработанное приложение "Калькулятор для расчета веса и количества саморезов и гвоздей" на языке программирования высокого уровня.

Чтобы определить какую технологию программирования использовать для написания программы, нужно проанализировать каждую из них:

Процедурное — метод, в соответствии с которым программы пишутся как перечни последовательно выполняемых команд.

Структурное, модульное — метод написания программ небольшими независимыми структурированными частями (модулями), каждый из которых связан с какой-либо процедурой или функцией. Результирующая программа организуется в виде совокупности взаимосвязанных по определенным правилам модулей.

Объектно-ориентированное (ООП) — метод, основанный на использовании концепции объекта, абстрагирующего конкретные его реализации в предметной области. При этом данные тесно связываются с выполняемыми над объектами процедурами;

Функциональное — метод, основанный на разбиении алгоритма решения задачи на отдельные функциональные модули, а также описании их связей и характера взаимодействия.

В данном случае наиболее правильным решением будет использовать объектно-ориентированный подход к программированию, так как программа имеет графический интерфейс и различные модули, к которым удобно применить парадигмы объектно-ориентированного программирования.

Для разработки программы можно использовать различные языки программирования и среды разработки. Далее проведем анализ каждого языка программирования, подходящего для разработки приложений под операционную систему windows, взвесив их плюсы и минусы.

Python — скриптовый язык общего назначения. Скриптовые языки обычно используются для небольших задач, но Python никак не вписывается в эти рамки. Это универсальный молодой язык.

Преимущества python:

1. Интерпретируемость. Это значит, что код в Python не переводится в машинный код, а выполняется программой-интерпретатором. Благодаря этому код в Python запустится на любой платформе, на которой установлен интерпретатор.
2. Поддержка различных парадигм программирования. Он поддерживает и функциональное, и структурное, и объектно-ориентированное программирование.
3. Простой минималистичный синтаксис: код легко писать, читать и поддерживать.
4. Большой выбор фреймворков.

Недостатки python:

1. Низкая скорость выполнения программ. Программы, написанные на Python, работают медленнее, чем аналогичные программы, разработанные на других языках.
2. Конвертация программы в exe. Программы, написанные на Питоне, имеют расширение “py”, и, чтобы использовать на Windows, их необходимо конвертировать в exe. После этого объём памяти, занимаемый программой, увеличивается в несколько раз.
3. Повышенное потребление памяти. Python — это язык, известный гибкостью подходов к типизации данных. Эта же динамическая типизация приводит к повышенному потреблению памяти.

C# — это язык программирования от компании Microsoft. Изначально его создавали для проектов под Windows.

Преимущества С#:

1. Платформа .NET Framework. Она умеет переводить код из понятного человеку в понятный процессору — то есть компилировать.

Преимущество платформы в том, что она может исполнить любую C#-команду на любом процессоре — а на Windows работает не меньше тысячи разных моделей. Если бы не .NET код пришлось бы компилировать отдельно под каждый процессор.

1. Управление памятью. В С# зачистка памяти происходит автоматически, программисту не надо следить за расходом памяти.
2. Мультиплатформенность. Используя различные фреймворки, можно запускать код на различных операционных системах и платформах.
3. Высокая производительность, потому что он не накладывает никакой избыточной нагрузки на программу, не использующую какие-либо возможности.

Недостатки с#:

1. Скорость. Когда мы запускаем программу на C#, код исполняется не сразу, а сначала адаптируется под нужное “железо”.
2. Система требовательна к железу.

Java — мультифункциональный объектно-ориентированный язык со строгой типизацией.

Преимущества Java:

1. Понятный синтаксис
2. Работает на всех операционных системах. Благодаря Java-машине.
3. Можно писать ПО любой сложности, от маленьких утилит до огромных программных комплексов с миллионами строк кода.

Недостатки Java:

1. Сложно писать даже достаточно простые вещи. Например, чтобы прочитать файл по URL-ссылке в python понадобиться одна строчка кода, а в Java 5-10.
2. Код на Java получается громоздким и многословным
3. Часто программы на Java уступают по скорости программам, написанным на С#

Таким образом, анализ предметной области показывает, что самым оптимальным выбором языка программирования будет – С#. Так как программа будет написана для системы windows, с# предоставляет самый большой набор инструментов для удобной разработки.

Выбранный язык программирования поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм.

Ниже разберем каждое понятие подробнее:

Инкапсуляция - процесс разделения элементов абстракций, определяющих ее структуру (данные) и поведение (методы); инкапсуляция предназначена для изоляции контрактных обязательств абстракции (протокол/интерфейс) от их реализации. На практике это означает, что класс должен состоять из двух частей: интерфейса и реализации.

Наследование - концепция объектно-ориентированного программирования, согласно которой абстрактный тип данных может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов программного обеспечения.

Полиморфизм — это способность объекта использовать методы производного класса, который не существует на момент создания базового.

Далее нужно определиться со средой разработки. Для выбранного языка программирования самой оптимальной средой разработки будет Visual Studio 2022. Данная среда разработки содержит необходимые для работы плагины, обеспечивающие работу с .NET и графическим интерфейсом программы.

# **Определение структуры программного продукта**

## **Анализ процесса обработки информации и выбор структур данных для ее хранения**

К системе необходимо предъявить ряд требований, выполнение которых обеспечит достижение оптимальных значений всех параметров целевой функции.

Для хранения информации была выбрана структура данных типа массив. Массив — это структура данных с фиксированным и упорядоченным набором однотипных элементов (компонентов). Доступ к какому-либо из элементов массива осуществляется по имени и номеру (индексу) этого элемента. Количество индексов определяет размерность массива.

Так как программа не имеет сложных данных в больших объемах, массива будет достаточно для хранения и обработки информации.

Определим следующие переменные в программе:

|  |  |
| --- | --- |
| Тип и название | Назначение |
| string[] sizeForWood | Хранение массива размеров для вида «Саморезы по дереву» |
| string[] sizeListForRoofing |  |
| string[] sizeForUni |  |
| string[] sizeForSecret |  |
| string[] sizeForMetal |  |
| string[] sizeForPSH |  |
| string[] sizeForGun |  |
| string[] sizeForNails |  |
|  |  |
|  |  |

## **Выбор методов решения задачи и разработка основных алгоритмов предметной области**

Задачами программы является:

1. Расчет количества гвоздей и саморезов, выбранного вида и размера. Исходные данные – введенный пользователем вес.
2. Расчет веса крепежных материалов, выбранного вида и размера. Исходные данные – введенное пользователем количество (в штуках).

В зависимости от задачи, выбранной пользователем программы будет реализован алгоритм действий и расчетов.

Таблица 1 – общий вариант выполнения приложения

|  |  |
| --- | --- |
| Действия исполнителя | Отклик программы |
| 1. Пользователь выбирает вид гвоздей или саморезов. | 1. Система отображает необходимые данные и выводит в поле «Размер» значения доступные для данного вида |
| 1. Пользователь выбирает размер крепежных материалов | 1. Система отображает необходимые данные |
| 1. Пользователь выбирает вид перевода (из килограмм в штуки или из штук в килограммы) | 1. Система отображает необходимые данные, а также меняет значения зависимых полей |
| 1. Пользователь вводит необходимое количество(штук или килограмм) в специальное поле | 1. Система отображает необходимые данные. |
| 1. Пользователь кликает на кнопку расчета | 1. Программа выбирает необходимый алгоритм подсчета и выводит результат на экран. |

Альтернатива 1:

1.Пользователь вводит новое выражение, или изменяет старое.

2.Система переходит к пункту 8 типичного хода событий.

Альтернатива 2:

1.Пользователь выбирает возврат к варианту «Вид» и изменяет данные.

2. Система переходит к пункту 2 типичного хода событий

Альтернатива 3:

1. Пользователь выбирает возврат к варианту «Размер» и изменяет данные.
2. Система переходит к пункту 4 типичного хода событий

Альтернатива 4:

1. Пользователь выбирает возврат к пункту «Тип перевода» и изменяет данные
2. Система переходит к пункту 6 типичного хода событий.

Рассмотрим каждый вариант работы программы более подробно.

Таблица 2 – вариант работы программы при выборе типа перевода «из килограмм в штуки»

|  |  |
| --- | --- |
| Действия исполнителя | Отклик программы |
| 1. Пользователь выбирает вид гвоздей или саморезов. | 1. Система отображает необходимые данные и выводит в поле «Размер» значения доступные для данного вида |
| 1. Пользователь выбирает размер крепежных материалов | 1. Система отображает необходимые данные |
| 1. Пользователь выбирает вид перевода(из килограмм в штуки или из штук в килограммы) | 1. Система отображает необходимые данные, а также меняет значения зависимых полей |
| 1. Пользователь вводит необходимое количество веса в килограммах. | 1. Система отображает необходимые данные и выбирает массив значений для дальнейшего расчета. |
| 1. Пользователь кликает на кнопку расчета | 1. Программа выполняет необходимый алгоритм подсчета и выводит результат на экран. |

Данный алгоритм показывает, как программа будет работать при выборе пользователем варианта «Перевод из килограмм в штуки».

Таблица 3 - вариант работы программы при выборе типа перевода «из штук в килограммы»

|  |  |
| --- | --- |
| Действия исполнителя | Отклик программы |
| 1. Пользователь выбирает вид гвоздей или саморезов. | 1. Система отображает необходимые данные и выводит в поле «Размер» значения доступные для данного вида |
| 1. Пользователь выбирает размер крепежных материалов | 1. Система отображает необходимые данные |
| 1. Пользователь выбирает вид перевода(из килограмм в штуки или из штук в килограммы) | 1. Система отображает необходимые данные, а также меняет значения зависимых полей |
| 1. Пользователь вводит необходимое количество веса в килограммах. | 1. Система отображает необходимые данные и выбирает массив значений для дальнейшего расчета. |
| 1. Пользователь кликает на кнопку расчета | 1. Программа выполняет необходимый алгоритм подсчета и выводит результат на экран. |

Альтернативы соответствуют альтернативам общего алгоритма приложения.

Данный алгоритм отображает работу программы при выборе пользователем варианта «перевод из штук в килограммы». При подсчете используются нормы ГОСТов для каждого крепежного материала.

## Построение структурной схемы программного продукта

Структурной называют схему, отражающую состав и взаимодействие по управлению частей разрабатываемого ПО.

Структурными компонентами программной системы или программного комплекса могут служить программы, подсистемы, базы данных, библиотеки ресурсов. Структурная схема программного комплекса демонстрирует передачу управления от программы-диспетчера (главной программы) соответствующей программе(методу).

Следующая схема отражает структуру и взаимодействие компонентов программного продукта «программы расчета веса и количества

саморезов и гвоздей»:

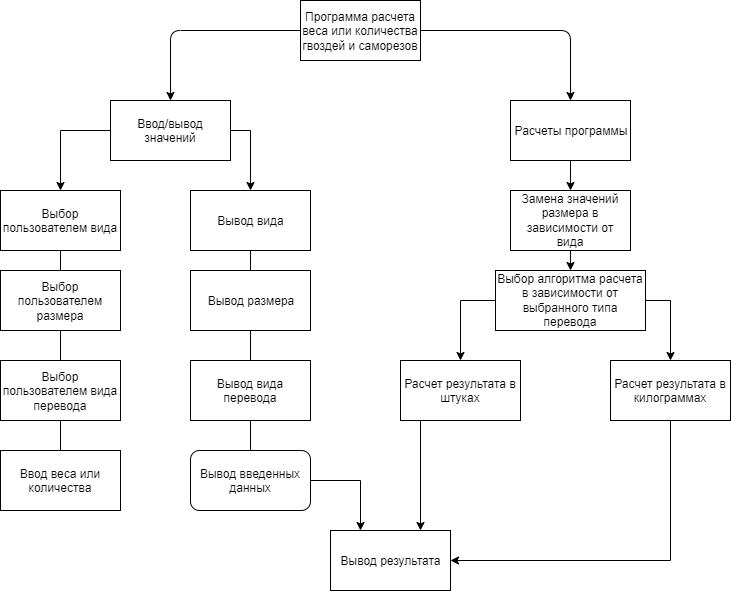


Рисунок 1 – структурная схема программного продукт

# **3. Разработка интерфейса пользователя**

## **3.1 Построение диаграммы вариантов использования**

UML — это сокращение от Unified Modeling Language, он является стандартизированным языком моделирования, состоящим из интегрированного набора диаграмм, разработанных, чтобы помочь разработчикам систем и программного обеспечения в определении структуры и взаимодействии пользователя с системой.

UML представляет собой набор лучших инженерных практик, которые доказали свою эффективность в моделировании больших и сложных систем и является очень важной частью разработки объектно-ориентированного программного обеспечения.

UML использует в основном графические обозначения, чтобы выразить дизайн программных проектов. Использование UML помогает проектным группам общаться, изучать потенциальные проекты и проверять архитектурный дизайн программного обеспечения.

UML-диаграммы - это способ наглядно описать архитектуру, проектирование и реализацию комплексных программных систем.

Диаграммы вариантов использования показывают, как пользователи(агенты) взаимодействуют с системой. Они служат обобщенной моделью связей между агентами и системами и объясняют устройство системы для аудитории без особой технической подготовки.

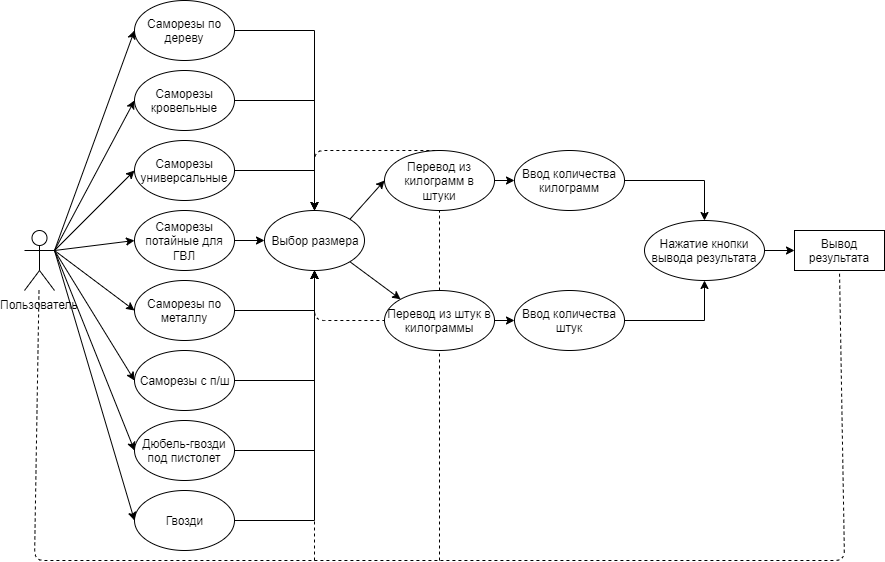


Рисунок 2 – Диаграмма вариантов использования

Таблица 4 – Условные обозначения для UML-диаграммы

|  |  |
| --- | --- |
| Условное обозначение | Пояснение |
|  | Агент(пользователь) приложения |
|  | Вариант взаимодействия пользователя с системой |
|  | Входные/выходные данные |
|  | Альтернативные варианты использования |
|  | Основные варианты использования |

## **Разработка форм ввода-вывода информации**

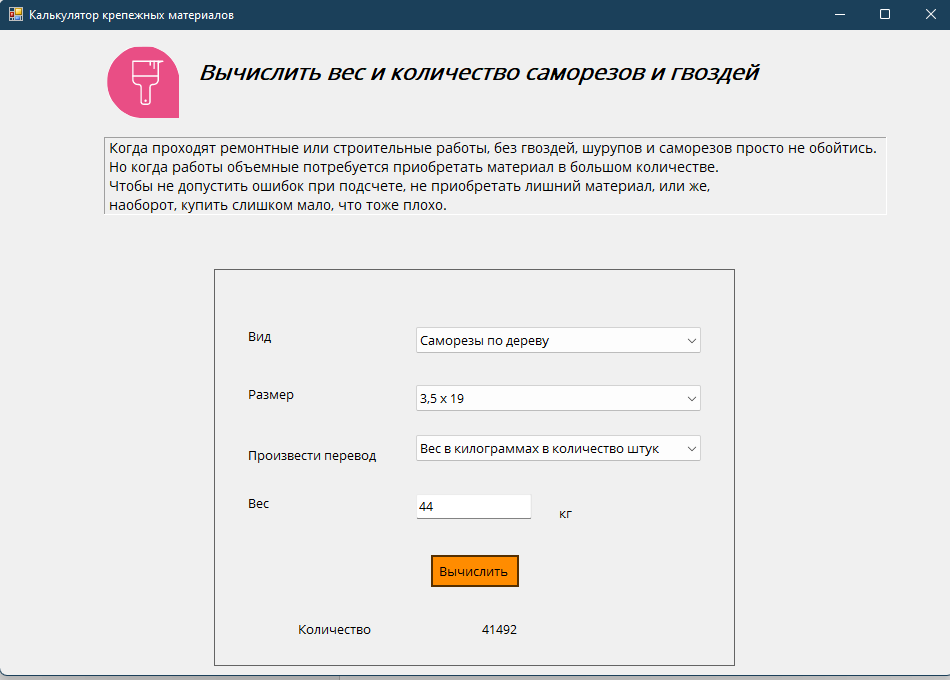


Рисунок 3 – общий вид пользовательской формы

Программа имеет одну пользовательскую форму, в которой происходит всё взаимодействие пользователя с программой. Форма является главной и содержит следующие поля:

* Заголовок приложения
* Описание
* Вид
* Размер
* Произвести перевод
* Вес/количество
* Кнопка «Вычислить»
* Поле вывода результата

Разберем каждое поле подробнее.

Поле «Вид» представлено в виде выпадающего списка видов саморезов и гвоздей. Пользователь не может редактировать значения поля, возможен только выбор одного из представленных значений.



Рисунок 4 – поле «Вид»

Поле «Размер» представлено в виде выпадающего списка размеров. Перечень размеров изменяется в зависимости от выбора в предыдущем поле «Вид». Редактирование поля пользователем невозможно. Возможен выбор готового значения.



Рисунок 5 – поле «Размер»

Поле «Произвести перевод» также представлено в виде выпадающего списка, состоящего из двух значений «Вес в килограммах в количество штук» и «Количество штук в килограммы».



Рисунок 6 – поле «Произвести перевод»

Поле «Вес»/ «Количество» представлено в виде строки для ввода пользовательских значений. Оно может принимать только числовые значения. Подпись к полю изменяется в соответствии с выбором пользователя в поле «Произвести перевод».



Рисунок 7 – поле «Вес»/ «Количество»

Кнопка «Вычислить» запускает метод, вычисляющий итоговое значение по формуле необходимой для выбранной пользователем совокупности значений.

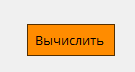


Рисунок 9 – кнопка «Вычислить»

Поле вывода результата имеет подпись «Вес» или «Количество». Подпись зависит от значения выбранного в поле «Произвести перевод», если выбрано значение «Вес в килограммах в количество штук», то значением подписи становится текст «Количество:», если выбрано значение «Количество штук в килограммы», то значением подписи становится текст «Вес:».



Рисунок 10 – поле вывода результата

# **Разработка диаграммы классов программы**

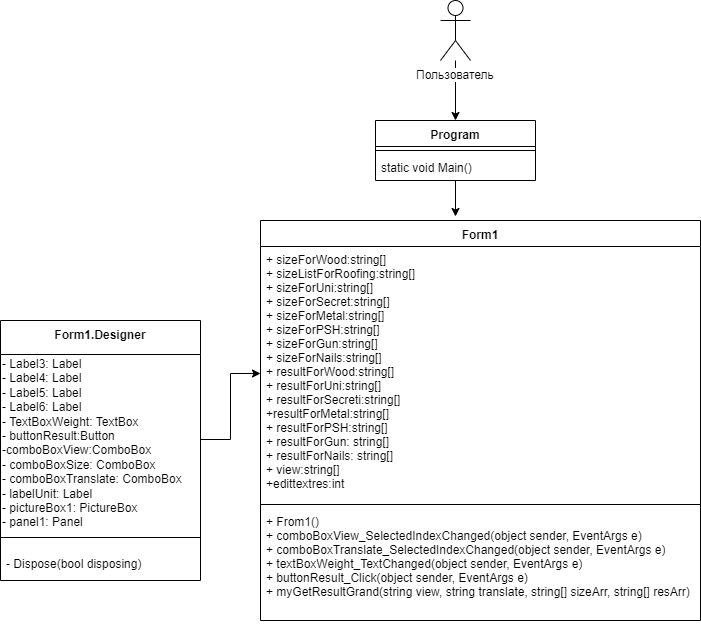


Рисунок 11 – UML диаграмма классов

Программа состоит из трёх основных классов. После запуска программы пользователем запускается класс Program, в котором вызывается метод Main(), вызывающий класс Form1, и запускающий графическую форму приложения.

Класс Form1 содержит следующие поля: sizeForWood:string[],sizeListForRoofing:string[],sizeForUni: string[], sizeForSecret:string[], sizeForMetal: string[], sizeForPSH:string[],sizeForGun:string[], sizeForNails:string[] – данные поля содержат в себе массив значений размеров для различных типов саморезов и гвоздей.

resultForWood:string[], resultForUni:string[],resultForSecret:string[], resultForMetal:string[], resultForPSH:string[], resultForGun:string[],resultForNails:string[] – данные поля содержат в себе массив значений для вычисления конечного результата для каждого из видов гвоздей и саморезов.

view:string[] – поле содержащее в себе массив со значениями видов саморезов и гвоздей.

edittextres: int – поле предназначено для хранения числа веденного пользователем с клавиатуры.

Класс Form1.Designer содержит поля, инициализирующие поля формы.

# **Разработка диаграммы последовательности**