Министерство образования и науки РФ

ФГАОУ ВО «Уральский федеральный университет   
имени первого Президента России Б. Н. Ельцина»

Институт новых материалов и металлургии

Кафедра «Теплофизика и информатика в металлургии»

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись расшифровка подписи

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись расшифровка подписи

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**Разработка программного обеспечения**

**«****Калькулятор для строительства частных домов»**

**«Введение в технологию разработки программного обеспечения»**

Направление 09.03.02 «Информационные системы и технологии»   
(уровень бакалавриата)

Образовательная программа 09.03.02/03.01   
«Информационные системы и технологии в металлургии»

Руководитель

ст. преподаватель, к.т.н. А. С. Истомин

должность, звание подпись расшифровка подписи

Нормоконтролер

ст. преподаватель, к.т.н. А. С. Истомин

должность, звание подпись расшифровка подписи

Студенты:

НМТ-213901 Ю. А. Коротков

номер группы подпись расшифровка подписи

НМТ-213901 М.Е.Назаров

номер группы подпись расшифровка подписи

Екатеринбург

2023

**РЕФЕРАТ**

Пояснительная записка изложена на 29 листах, содержит 31 рисунок и 2 приложения.

РАСЧЕТ ОПТИМАЛЬНЫХ СТОИМОСТЕЙ УСЛУГ ДЛЯ СТРОИТЕЛЬСТВА ДОМА, ПРОГРАММИРОВАНИЕ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, АРХИТЕКТУРА, ИНТЕРФЕЙС, VISUAL STUDIO C#, ИНСТАЛЛЯЦИЯ.

Проект по модулю посвящен разработке программного обеспечения «Калькулятор для строительства частных домов».

Отражены основные этапы разработки программного обеспечения: постановка задачи, реализация тестового варианта расчета в электронных таблицах Microsoft Excel; проектирование и реализация программного средства – математической библиотеки и пользовательского интерфейса; разработка системы автоматизированного тестирования математической библиотеки; создание справочной документации; подготовка дистрибутива. Размещение исходного программного кода выполнено в системе удаленного контроля версий GitHub.

Основными функциями программного обеспечения является расчет оптимальных стоимостей услуг для строительства дома.

Основные конечные пользователи программного обеспечения – строительные организации.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc137146717)

[1 ПОСТАНОВКА ЗАДАЧИ, ПРОВЕРКА КОРРЕКТНОСТИ АЛГОРИТМА РАСЧЕТА И ОРГАНИЗАЦИЯ РАБОТЫ НАД ПРОЕКТОМ В СИСТЕМЕ YOUTRACK 5](#_Toc137146718)

[1.1 Физическая постановка задачи 5](#_Toc137146719)

[1.2 Математическая модель задачи 5](#_Toc137146720)

[1.3 Создание тестового варианта расчета в электронных таблицах Microsoft Excel 6](#_Toc137146721)

[1.4 Постановки задач для выполнения отдельных этапов проекта в системе YouTrack 6](#_Toc137146722)

[2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 8](#_Toc137146723)

[2.1 Разработка архитектуры системы 8](#_Toc137146724)

[2.2 Разработка блок-схемы работы пользователя с программой 11](#_Toc137146725)

[2.3 Создание программного обеспечения в системе управления версиями GitHub 11](#_Toc137146726)

[2.4 Разработка математической библиотеки 13](#_Toc137146727)

[2.5 Реализация пользовательского интерфейса 14](#_Toc137146728)

[2.6 Обработка исключительных ситуаций 15](#_Toc137146729)

[2.7 Создание справочной помощи 15](#_Toc137146730)

[2.8 Создание дистрибутива 16](#_Toc137146731)

[3 РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ МАТЕМАТИЧЕСКОЙ БИБЛИОТЕКИ 20](#_Toc137146732)

[3.1 Проектирование системы 20](#_Toc137146733)

[3.2 Реализация системы 20](#_Toc137146734)

[4 ОПИСАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 21](#_Toc137146735)

[4.1 Установка и настройка программного средства 21](#_Toc137146736)

[4.2 Функциональные возможности программного продукта 21](#_Toc137146737)

[4.3 Технология выполнения расчетов 23](#_Toc137146738)

[ЗАКЛЮЧЕНИЕ 24](#_Toc137146739)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 25](#_Toc137146740)

[ПРИЛОЖЕНИЕ 1 ЛИСТИНГ КОДА ТЕСТОВ 26](#_Toc137146741)

[ПРИЛОЖЕНИЕ 2 ЛИСТИНГ КОДА РАЗМЕТКИ СТРАНИЦЫ 28](#_Toc137146742)

# ВВЕДЕНИЕ

Целью данного проекта является разработка приложения «Калькулятор для строительства частных домов», которое поможет пользователям рассчитывать стоимость постройки нового здания, показав затраты на каждый пункт строительства, что в итоге поможет рассчитывать бюджет.

Для достижения этой цели необходимо решить следующие задачи:

1. Разработать интерфейс приложения, который будет интуитивно понятен и удобен для использования.
2. Реализовать алгоритмы расчета общей стоимости строительных материалов, стоимости работы, стоимости всех разрешений, общей стоимости отделочных материалов, общей стоимости доп. опций, общей стоимости строительства, а также ежемесячный платеж по кредиту, расчёт необходимых накоплений для оплаты строительства + резервный фонд и сколько нужно откладывать каждый месяц чтобы скопить сумму к началу строительства.
3. Обработать возможные исключительные ситуации, которые могут возникнуть во время работы приложения.

Для реализации данного проекта будет использоваться язык программирования C# и среда разработки Visual Studio. В процессе разработки приложения будут применяться современные подходы и методологии разработки, такие как Agile и Scrum.

В результате выполнения проекта будет создано полноценное приложение, которое поможет строительным организациям в расчётах, поможет быстрее согласовывать план строительства, а также быстро его корректировать, если заказчик того пожелает.

# 1 ПОСТАНОВКА ЗАДАЧИ, ПРОВЕРКА КОРРЕКТНОСТИ АЛГОРИТМА РАСЧЕТА И ОРГАНИЗАЦИЯ РАБОТЫ НАД ПРОЕКТОМ В СИСТЕМЕ YOUTRACK

## 1.1 Физическая постановка задачи

Необходимо разработать приложение, которое будет рассчитывать стоимость постройки дома. Основной задачей программы будет расчет необходимого количество накоплений и сумму, которую нужно откладывать каждый месяц для накопления такой суммы к моменту начала строительства дома.

Для расчета стоимости дома программа должна запрашивать у пользователя количество этажей дома, количество доп.опций и площади этажей, а затем проводить необходимые вычисления.

На основе полученных данных программа рассчитает отдельно общую площадь дома, общую стоимость разрешений и материалов

Важной функцией программы будет также возможность сохранения результатов расчетов в виде отчета в формате .pdf и .jpg.

## 1.2 Математическая модель задачи

* Для расчёта общей площади программа будет складывать площади всех этажей.
* Для расчета стоимости разрешений программа будет складывать стоимости “разрешения на строительство” и “стоимость оформления остальных разрешений и бумаг” (разрешение на терраформирование, временное пользование дороги тяжелой техникой и т. д.)
* Для определения стоимости материалов программа будет умножать площадь дома на стоимость (строительных+отделочных) материалов
* Для определения общей стоимости строительства программа будет складывать всё нижеперечисленное, а также стоимость установки доп.опций и оплаты рабочим
* Для расчёта минимальных накоплений общая стоимость умножается на 1.2, где две десятые доли – резервный фонд
* Так же программа рассчитает какой будет ежемесячный платёж если не копить деньги, а брать кредит по формуле ((общая стоимость \* процент по кредиту / количество месяцев стройки) \*100)/100
* Кроме того программа рассчитает какую сумму необходимо откладывать каждый месяц для накопления на оплату строительства + резервный фонд: ((общая стоимость+резерв/кол-во месяцев)\*100)/100
* Для сохранения результатов расчетов пользователя программа будет конвертировать pdf-файл, который будет хранить информацию о пользователе и все запрашиваемые расчеты.

## 1.3 Создание тестового варианта расчета в электронных таблицах Microsoft Excel

В электронных таблицах Microsoft Excel создана эмуляция расчетов по заданным параметрам (Рис. 1). В дальнейшем эти данные использованы для тестов в качестве достоверного источника.

Изображение выглядит как текст, снимок экрана, линия, программное обеспечение

Автоматически созданное описание

Рисунок 1 – расчеты по заданным параметрам в Microsoft Excel

## 1.4 Постановки задач для выполнения отдельных этапов проекта в системе YouTrack

YouTrack — это инструмент управления проектами, который легко адаптируется под любые процессы пользователя: планирование проектов и отслеживайте задач, использование Agile-досок, организация спринтов и релизов, ведение базы знаний, использование отчетов и панели мониторинга, создание рабочих процессов.

Переходим в раздел «Проекты» и создаем новый проект Scrum «HouseBuilder» (Рис. 2). Методология Scrum помогает настроить рабочий процесс и управлять им.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рисунок 2 – проект «HouseBuilder» в системе YouTrack

Agile-доски — инструмент для совместной работы, в его основе лежат принципы Scrum-методологии, данный подход широко используется в отраслях с постоянным потоком задач.

Переходим в Agile-доску нашего проекта, создаем спринты и добавляем туда все наши задачи, назначаем исполнителей и приоритет (Рис. 3 – 6).

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3 – заполнение первого спринта Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 4 – заполнение второго спринта Изображение выглядит как текст, программное обеспечение, число, Значок на компьютере

Автоматически созданное описание

Рисунок 5 – заполнение третьего спринта Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 6 – заполнение четвертого спринта

Ссылка на проект: [https://trunzy.youtrack.cloud/projects/d1a5eced-7ca4-4b8c-b505-6e11b11483c7](https://vk.com/away.php?to=https%3A%2F%2Ftrunzy.youtrack.cloud%2Fprojects%2Fd1a5eced-7ca4-4b8c-b505-6e11b11483c7&cc_key=)

# 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 2.1 Разработка архитектуры системы

Для разработки архитектуры системы для программного обеспечения «Builder» необходимо определить ее основные проекты и их взаимодействие.

1. Библиотека. Этот проект содержит в себе класс, в котором определены вычисляемые поля, хранящие в себе информацию для отображения пользователю (Рис. 7).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 7 – код класса библиотеки

1. Тесты. Этот проект предназначен для тестирования проектной библиотеки, описанной выше. Тесты проверяют корректность выводимых значений и обработку исключительных ситуаций. Листинг кода тестов представлен в Приложении 1.
2. WPF. Этот проект предназначен для предоставления пользователю возможности комфортного пользования библиотекой. Проект состоит из компонентов, представленных ниже.
   1. ViewModel. Класс, реализующий связь между графическим интерфейсом и библиотекой. В нашем случае класс – BuilderViewModel, а также вспомогательная сущность FloorGeneratorModel, которая служит для создания элементов списка. BuilderViewModel представлен на рисунке 8. FloorGeneratorModel представлен на рисунке 9. Для их создания используется библиотека ReactiveUI.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, снимок экрана

Автоматически созданное описание

Рисунок 8 - класс BuilderViewModel

Изображение выглядит как снимок экрана, текст, программное обеспечение, линия

Автоматически созданное описание

Рисунок 9 - класс FloorGeneratorModel

* 1. Графический интерфейс.Класс, реализующий графический интерфейс. В нашем случае класс – MainWindow.xaml, код которого представлен в приложени 2. Для создания связи между графическим интерфейсом и ViewModel используется библиотека ReactiveUI. Графический интерфейс создан при помощи компонентов графической библиотеки. График был создан при помощи специализированной библиотеки LiveCharts.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 10 – графический интерфейс

* 1. Конверторы значений. Два класса, предназначенных для конвертации входных данных от пользователя из строки в специфический тип (Рисунок 11–12).

Изображение выглядит как текст, снимок экрана, программное обеспечение, линия

Автоматически созданное описание

Рисунок 11 - класс DoubleToStringConverter

Изображение выглядит как текст, программное обеспечение, линия, снимок экрана

Автоматически созданное описание

Рисунок 12 - класс UintToStringConverter

## 2.2 Разработка схемы работы пользователя с программой

Пользователь взаимодействует с графическим интерфейсом, ViewModel в свою очередь считывает эти изменения при помощи интерфейса INotifyPropertyChanged, реализованного в классе ReactiveObject библиотеки ReactiveUI. VM использует библиотеку для расчетов и выводит полученные данные на интерфейс. Таким же образом формируется и отчеты в виде .pdf и .jpg файлов.

Схема работы пользователя с программой представлена на рисунке 13.

Изображение выглядит как снимок экрана, текст, диаграмма, линия

Автоматически созданное описание

Рисунок 13 – схема работы пользователя с программой

## 2.3 Создание программного обеспечения в системе управления версиями GitHub

Создан локальный репозиторий Git и внешний репозиторий GitHub (Рис. 14). Работа с репозиторием велась при помощи приложения GitHub Desktop (Рис. 15). Также создан файл .gitignore.

Git – это распределенная система контроля версий с открытым исходным кодом, которую можно использовать для отслеживания изменений в файлах.

GitHub – это веб-интерфейс, в котором можно хранить репозитории Git, а также эффективно отслеживать и управлять изменениями. С его помощью разные разработчики имеют доступ к коду одного проекта, есть возможность вносить свои изменения в проект одновременно с другими разработчиками.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 14 - внешний репозиторий Git

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 15 - приложение GitHub Desktop

## 2.4 Разработка математической библиотеки

Математическая библиотека будет использоваться для выполнения математических операций в программе и содержать в себе функции для расчета ИМТ, оптимальных показателей, а также других математических операций, необходимых для работы программы.

В решении создаем проект библиотеки классов .NET Standart, в нем создаем класс BuilderCalculator с конструктором, принимающим 4 аргумента - вес, рост, возраст и стиль жизни, и с вычислительными полями для каждого из требуемых параметров.

Код класса математической библиотеки представлен на рисунке 16.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 16 - код класса математической библиотеки

## Реализация пользовательского интерфейса

В решение добавляем проект WPF на .NET 6.0, в ссылки добавляем проект библиотеки, добавляем соответствующий using. Далее реализуем ViewModel (BuilderCalculatorViewModel). Реализуем модель при помощи библиотеки ReactiveUI, в которой есть удобная реализация интерфейса INotifyPropertyChanged.

Следующим шагом создаем графический интерфейс для пользователя. Сам интерфейс реализуем на языке xaml. XAML — это язык разметки, который позволяет описывать элементы интерфейса в виде дерева объектов. XAML-разметка представлена в приложении 2. График для интерфейса выполняем с помощью библиотеки LiveCharts.WPF.

Так же создаем ряд вспомогательных файлов-классов для работы с интерфейсом, к примеру - два конвертера значений (описанных в пункте 2.1) и метод для создания отчетов FastReports. Код метода представлен на рисунке 17.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 17 – класс для создания FastReport

## 2.6 Обработка исключительных ситуаций

Обработка исключительных ситуаций, по типу некорректного ввода пользователя или любой другой ошибки, происходит благодаря блоку try-catch. В проекте математической библиотеки кидается ошибка при некорректном вводе пользователя (значения роста, веса и т. д. меньше или равны 0). В блоке try – написан код, который выполняется в штатной ситуации, в catch – выводится окно (MessageBox), сигнализирующий об ошибке с её полным текстом.

Таким образом, при возникновении ошибки в процессе работы приложение не «падает», а продолжает свою работу, ожидая более верных данных от пользователя.

## 2.7 Создание справочной помощи

Было создано окно справочной помощи, оно открывается на главном окне приложения при помощи кнопки «Справка» (Рис. 18).

Изображение выглядит как текст, электроника, снимок экрана, Шрифт

Автоматически созданное описание

*Рисунок 18 – окно справочной помощи*

## 2.8 Создание дистрибутива

Для создания дистрибутива для получившегося приложения создаем отдельный проект (Setup1) в нем задаем все необходимые параметры, добавляем все файлы (.dll, .exe, .frx) и создаем ярлык с кастомной иконкой (Рисунок 19). После этого выполняем сборку всего решения и тестов удаления (Рисунок 20) и установки программы через мастер-установщик (Рисунок 21–25).

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Рисунок 19 – проект SetUp

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 20 – подготовка к удалению

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 21 – мастер установки

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 22 – выбор папки для установки

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 23 – подтверждение установки

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 24 – процесс установки

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 25 – завершение установки

# 3 РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ МАТЕМАТИЧЕСКОЙ БИБЛИОТЕКИ

## 3.1 Проектирование системы

Cпроектированы модульные тесты для математической библиотеки, предусматривающие возможность проверить работоспособность и корректность ее работы.

## 3.2 Реализация системы

Для отлаживания библиотеки, описанной ранее (пункт 2.4), добавляем проект с Unit тестами (NUnit). В него так же добавляем ссылку на проект библиотеки. Модульное тестирование предусматривает проверку реакции на некорректные введенные данные и корректность выходных значений.

Пример теста представлен на рисунке 26, а вывод ошибки из-за ввода некорректных данных на рисунке 27. Листинг кода всех тестов представлен в приложении 1.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 26 – пример теста

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 27 – вывод ошибки из-за некорректных входных данных

# 4 ОПИСАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 4.1 Установка и настройка программного средства

Для установки программного продукта достаточно следовать инструкциям мастера-установщика. Все необходимые для работы файлы будут помещены в папку с приложением (Рисунок 28).

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 28 – необходимые файлы в папке с приложением

## 4.2 Функциональные возможности программного продукта

Функциональными возможностями программного продукта являются расчет и структурированный вывод на экран информации, построение графика для визуализации данных (Рисунок 29) и возможность сохранения данных в виде .pdf, .jpg отчетов, генерирующихся программно (Рисунок 30). Отчеты будут сохранены в: C\Users\"Пользователь"\AppData\Roaming\reportFolder (Рисунок 31).

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание

Рисунок 29 – график на основе рассчётов

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 30 – pdf-отчет

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 31 – генерация pdf и .jpg-файлов

## 4.3 Технология выполнения расчетов

Для решения задачи использование сложных подходов, таких как асинхронность или многопоточность, не требуется, поскольку задача связана только с примитивными cpu bound операциями и ее выполнение не зависит от внешних запросов и операций ввода-вывода. Применение этих подходов в данном контексте скорее всего потребует больше времени и ресурсов на создание и поддержку дополнительных потоков, что может снизить общую производительность системы.

# ЗАКЛЮЧЕНИЕ

В данной работе описаны цели и задачи проекта, а также методы и инструменты, использованные для его реализации.

В ходе выполнения проекта разработано приложение «Калькулятор для строительства частных домов» на языке C# с использованием среды разработки Visual Studio и с применением Agile и Scrum методологий; разработан интерфейс приложения, который интуитивно понятен и удобен для использования; реализованы алгоритмы расчета общей стоимости строительных материалов, стоимости работы, стоимости всех разрешений, общей стоимости отделочных материалов, общей стоимости доп. опций, общей стоимости строительства, а также ежемесячный платеж по кредиту, расчёт необходимых накоплений для оплаты строительства + резервный фонд и сколько нужно откладывать каждый месяц чтобы скопить сумму к началу строительства – для понимания и удобства заказчика, чтобы он мог согласовать план или чтобы команда застройщика быстро могла подкорректировать план строительства так, чтобы заказчика удовлетворяли и план строительства, и его стоимость.

Разработанное приложение может помочь строительным организациям в расчётах, поможет быстрее согласовывать план строительства, а также быстро его корректировать, если заказчик того пожелает. Таким образом, оно будет полезным инструментом для любой строительной организации, занимающейся постройкой частных домов.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Основы проектирования информационных систем. Учебное пособие. – СПб: Университет ИТМО, 2015. – 206 с.
2. Горлушкина Н.Н. Системный анализ и моделирование информационных процессов и систем. Учебное пособие. – СПб: Университет ИТМО, 2016. – 120 с.
3. Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. – Проектирование информационных систем: учебное пособие / 2-е изд., испр. – М.: Интернет-Университет информационных технологий (ИНТУИТ.РУ): БИНОМ. Лаборатория знаний, 2010. – 299 c.
4. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5. 6-е изд. – М.: ООО «И.Д. Вильямс», 2015. – 1312 с.
5. Флёнов М.Е. Библия С#. 2-е изд. – СПб.: БХВ-Петербург, 2011. – 560 с.
6. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. Практическое руководство - Санкт-Петербург, 2012. – 928 с.
7. Шилдт Г. Полное руководство С# 4.0 [Текст]: учебное пособие / Г. Шилдт — пер. с англ. Берштейн И. В. — Москва: Вильямс, 2012.— 1051 с.
8. Хайруллин Р.С. Программирование на C#: учебное пособие / Р.С. Хайруллин. – Казань: Изд-во Казанск. гос. архитект.-строит. ун-та, 2017. – 153 с.
9. Фаронов В. В. Создание приложений с помощью C#: Руководство программиста [Текст]: учебник / В. В. Фаронов — Москва: Эксмо, 2008. — 576 с.
10. Лавров В.В. Методические указания к выполнению курсовой работы для студентов бакалавриата, обучающихся по направлению «Информационные системы и технологии» и студентов магистратуры, обучающихся по программе «Информационные системы в металлургии» / В.В. Лавров, К.А. Щипанов, А.А. Бурыкин – Екатеринбург: УрФУ, 2014. – 49 с.
11. Лошкарев Н.Б. Указания к оформлению дипломных и курсовых проектов и работ: методические указания / Н.Б. Лошкарев, А.Н. Лошкарев, Л.А. Зайнуллин. – Екатеринбург: УГТУ–УПИ, 2007. – 49 с.

# ПРИЛОЖЕНИЕ 1 ЛИСТИНГ КОДА ТЕСТОВ

namespace TestProject1

{

public class Tests

{

[Test]

public void TestMinAll()

{

Builder Minall = new(1);

for (int i = 0; i < 5; i++)

Minall.FloorsArea.Add(-12);

Assert.Throws<ArgumentException>(() => Minall.CalculateTotalArea(), "Floors area cannot be negative");

}

[Test]

public void AreaGood()

{

Builder AreaGood = new(1);

for (int i = 0; i < 5; i++)

AreaGood.FloorsArea.Add(12);

double expected = 60;

double actual = AreaGood.CalculateTotalArea();

Assert.That(actual, Is.EqualTo(expected));

}

[Test]

public void OptionsGood()

{

Builder OptGood = new(1);

double expected = 5000;

double actual = OptGood.CalculateOptionCost();

Assert.That(actual, Is.EqualTo(expected));

}

[Test]

public void TotalAreaGood()

{

Builder TotalGood = new(1);

for (int i = 0; i < 2; i++)

TotalGood.FloorsArea.Add(12);

double expected = 24;

double actual = TotalGood.CalculateTotalArea();

Assert.That(actual, Is.EqualTo(expected));

}

/\*

This solution completely lacks the meaning of the tests given below

The problem is that constant variables in a real project would get the

value from a conditional database/directory, which currently do not exist.

\*/

[Test]

public void MaterialCost()

{

Builder MaterialCost = new(1);

double expected = 100000;

double actual = MaterialCost.CalculateMaterialCost();

Assert.That(actual, Is.EqualTo(expected));

}

[Test]

public void LaborCost()

{

Builder LaborCost = new(1);

double expected = 50000;

double actual = LaborCost.CalculateLaborCost();

Assert.That(actual, Is.EqualTo(expected));

}

[Test]

public void PermitCost()

{

Builder PermitCost = new(1);

double expected = 15000;

double actual = PermitCost.CalculatePermitCost();

Assert.That(actual, Is.EqualTo(expected));

}

[Test]

public void TotalCost()

{

Builder TotalCost = new(1);

for (int i = 0; i < 2; i++)

TotalCost.FloorsArea.Add(12);

double expected = 650000;

double actual = TotalCost.CalculateTotalCost();

Assert.That(actual, Is.EqualTo(expected));

}

[Test]

public void MonthlyPayment()

{

Builder MonthlyPayment = new(1);

for (int i = 0; i < 2; i++)

MonthlyPayment.FloorsArea.Add(12);

double expected = 271;

double actual = MonthlyPayment.CalculateMonthlyPayment();

Assert.That(actual, Is.EqualTo(expected).Within(0.5));

}

[Test]

public void TotalSavings()

{

Builder TotalSavings = new(1);

for (int i = 0; i < 2; i++)

TotalSavings.FloorsArea.Add(12);

double expected = 780000;

double actual = TotalSavings.CalculateTotalSavingsNeeded();

Assert.That(actual, Is.EqualTo(expected));

}

[Test]

public void MonthlySavings()

{

Builder MonthlySavings = new(1);

for (int i = 0; i < 2; i++)

MonthlySavings.FloorsArea.Add(12);

double expected = 32500;

double actual = MonthlySavings.CalculateMonthlySavings();

Assert.That(actual, Is.EqualTo(expected));

}

}

}

# ПРИЛОЖЕНИЕ 2 ЛИСТИНГ КОДА РАЗМЕТКИ СТРАНИЦЫ

<Window x:Class="HouseBuilderWindow.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:HouseBuilderWindow"

xmlns:vm="clr-namespace:HouseBuilderWindow.ViewModels"

xmlns:conv="clr-namespace:HouseBuilderWindow.Converters"

mc:Ignorable="d"

Title="MainWindow" Height="426" Width="714">

<Window.DataContext>

<vm:BuilderViewModel/>

</Window.DataContext>

<Window.Resources>

<conv:UintToStringConverter x:Key="uintToString"/>

<conv:DoubleToStringConverter x:Key="doubleToString"/>

</Window.Resources>

<StackPanel Orientation="Vertical">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="0 25 0 0">

<Label Content="Введите количество этажей для строющегося здания: " FontSize="16" Height="35" />

<TextBox x:Name="TextBox\_CountOfFloor" Width="100" Height="30" Text="{Binding FloorsCount, Mode=OneWayToSource, Converter={StaticResource uintToString}, UpdateSourceTrigger=PropertyChanged}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<StackPanel Orientation="Vertical" Width="300">

<StackPanel Orientation="Horizontal">

<Label Content="Количество доп. опций: "/>

<TextBox Text="{Binding NumberOfOptions, Mode=OneWayToSource, Converter={StaticResource uintToString}}" Width="25"/>

</StackPanel>

<ListView ItemsSource="{Binding FloorsGenerator}" >

<ListView.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal" Height="30">

<Label Content="{Binding Label, Mode=OneWay, UpdateSourceTrigger=PropertyChanged}" Width="200" Height="25" VerticalAlignment="Center"/>

<TextBox Width="50" Text="{Binding Area, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource doubleToString}}" Height="25" VerticalAlignment="Center"/>

</StackPanel>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackPanel>

<StackPanel Orientation="Vertical">

<StackPanel Orientation="Horizontal">

<Label Content="Общая площадь дома:" Width="325"/>

<Label Content="{Binding TotalArea, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Общая стоимость строительных материалов:" Width="325"/>

<Label Content="{Binding MaterialCost, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Стоимость работы:" Width="325"/>

<Label Content="{Binding LaborCost, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Стоимость всех разрешений:" Width="325"/>

<Label Content="{Binding PermitCost, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Общая стоимость отделочных материалов:" Width="325"/>

<Label Content="{Binding FinishingCost, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Общая стоимость доп. опций:" Width="325"/>

<Label Content="{Binding OptionCost, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Общая стоимость строительства:" Width="325"/>

<Label Content="{Binding TotalCost, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="ежемесячный платеж по кредиту:" Width="325"/>

<Label Content="{Binding MonthlyPayment, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Необходимое количество накоплений:" Width="325"/>

<Label Content="{Binding TotalSavingsNeeded, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Сумма отложения в месяц до начала строительства:" Width="325"/>

<Label Content="{Binding MonthlySavings, Mode=OneWay, Converter={StaticResource doubleToString}}"/>

</StackPanel>

</StackPanel>

</StackPanel>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="10">

<Button Height="20" Width="80" Content="Расчитать" Command="{Binding CountCommand}"/>

<Button Height="20" Width="130" Content="Сгенерировать отчет" Command="{Binding ReportCommand}" Margin="10 0 0 0"/>

<Button Height="20" Width="60" Content="Справка" Command="{Binding RefInfoCommand}" Margin="10 0 0 0"/>

</StackPanel>

</StackPanel>

</Window>