

Федеральное государственное автономное образовательное учреждение
высшего образования «Пермский национальный исследовательский
политехнический университет»

ОТЧЁТ

ПО ТВОРЧЕСКОЙ РАБОТЕ ЧАСТЬ 1

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «АРМ специалиста»

Выполнил:

студент первого курса

ЭТФ группы РИС-23-36

Акбашева Софья Руслановна

Проверила:

Доцент кафедры ИТАС О. А. Полякова

Пермь, 2024

РЕФЕРАТ

АРМ, SFML, СТРОИТЕЛЬСТВО, QT, КЛАСС, ОКНО, РАССЧЕТ, СОХРАНЕНИЕ ДАННЫХ.

Объектом исследования является рабочее место менеджера по продаже домов по финской технологии.

Предмет исследования — автоматизация рабочего места (АРМ) менеджера по продаже домов.

Целью работы является разработка калькулятора расчета стоимости строительства дома.

В результате проведенного исследования был разработан калькулятор, который позволяет рассчитать стоимость строительства дома.

ОГЛАВЛЕНИЕ

РЕФЕРАТ	2
ВВЕДЕНИЕ	5
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	6
2 ОПИСАНИЕ ИСПОЛЬЗУЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	7
3 РАЗРАБОТКА ПРОГРАММЫ	9
Анализ кода	9
4 ТЕСТИРОВАНИЕ ПРОГРАММЫ	13
ЗАКЛЮЧЕНИЕ	14
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	15
ПРИЛОЖЕНИЯ.....	16
ПРИЛОЖЕНИЕ А	16
Рисунок А.1 – uml диаграмма классов House и Prices	16
Рисунок А.2 – uml диаграмма класса Main_Window_1	17
Рисунок А.3 – uml диаграммы классов change_the_directory и save_data_window	17
ПРИЛОЖЕНИЕ Б.....	18
Листинг Б.1 – Файл house.h	18
Листинг Б.2 – Файл Prices.h.....	19
Листинг Б.3 – Файл main_window_1.h	20
Листинг Б.4 – Файл main_window_1.cpp	21
Листинг Б.5 – Файл change_the_directory.h	23
Листинг Б.6 – Файл change_the_directory.cpp	23
Листинг Б.7 – Файл save_data_window.h	25
Листинг Б.8 – Файл save_data_window.cpp.....	26
Листинг Б.9 – Файл other_fuctions.cpp	27
Листинг Б.10 – Файл main.cpp.....	28
Листинг Б.11 – Файл styles.css.....	29
Листинг Б.12 – Файл all_the_prices.txt	30
ПРИЛОЖЕНИЕ В	31
Рисунок В.1 – пользовательский интерфейс	31
Рисунок В.2 – расчёт стоимости строительства дома	32

Рисунок В.3 – сохранение данных	33
Рисунок В.4 – изменение цен	34
ПРИЛОЖЕНИЕ Г	35
Рисунок Г.1 – результат работы в GitHub.....	35

ВВЕДЕНИЕ

Целью данной работы является разработка калькулятора для расчета стоимости строительства дома по финской технологии.

Постановка задачи: разработать приложение, которое позволит рассчитать стоимость строительства дома, а также сможет сохранить полученный результат и, при необходимости, изменить цены.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) Анализ предметной области.
- 2) Описание используемого программного обеспечения.
- 3) Разработка программы.
- 4) Тестирование программы на корректность работы.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В настоящее время все больше людей задумываются о строительстве собственного дома. Однако, процесс строительства может быть достаточно сложным и затратным. Финская технология строительства предлагает инновационное решение этой проблемы. Она предполагает производство каркаса дома на заводе, а затем его сборку на участке из готовых панелей. Это значительно сокращает время строительства и уменьшает затраты [1].

Однако, для того чтобы рассчитать стоимость строительства дома по финской технологии, необходимо учитывать множество факторов. Например, площадь дома, количество этажей, материалы, используемые при строительстве, и многое другое. Для упрощения этого процесса можно использовать специальный калькулятор.

Калькулятор для расчета стоимости строительства дома по финской технологии будет полезен как для строительных компаний, так и для частных лиц. Он позволит быстро и точно определить стоимость строительства, учитывая все необходимые параметры.

Для создания такого калькулятора необходимо провести анализ предметной области. Это включает изучение всех возможных параметров, которые могут влиять на стоимость строительства, а также разработку алгоритмов для их учета. Кроме того, важно учесть особенности финской технологии строительства и ее преимущества перед другими методами.

После проведения анализа можно приступить к разработке самого калькулятора. Он должен быть простым в использовании, интуитивно понятным и предоставлять точные результаты.

Таким образом, создание калькулятора для расчета стоимости строительства дома по финской технологии является актуальной задачей. Его использование поможет существенно упростить процесс строительства и сделать его более доступным для широкого круга людей.

2 ОПИСАНИЕ ИСПОЛЬЗУЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В работе был использован язык программирования C++. C++ — это язык программирования, который был разработан в 80-х годах прошлого века как расширение языка C. Этот язык отличается от Си тем, что имеет больший набор возможностей, включая объектно-ориентированное программирование и шаблоны.

C++ используется для создания программного обеспечения разного рода: от игр до операционных систем. Этот язык также широко применяется в интенсивной обработке данных и научных расчетах.

C++ предоставляет разработчикам мощный и гибкий инструмент для создания программного обеспечения. Он позволяет писать эффективный и быстрый код, что делает его одним из наиболее популярных языков программирования в мире [2].

Qt — библиотека C++, которая предоставляет разработчикам программный интерфейс для создания графического интерфейса пользователя (GUI) и работает почти на всех известных платформах. Включает в себя набор инструментов для работы с сетью, отображения 2D- и 3D-графики, обработки изображений, XML, SQL, thread-safety, Unicode и интернационализации. Является кроссплатформенной и открытой. Написана на языках программирования C++ и QML. Используется для создания приложений с графическим интерфейсом пользователя, таких как компьютерные игры, различные редакторы, пользовательские приложения для Linux, такие как KDE Plasma, многие программы для Windows, такие как Opera, Skype, VirtualBox, Amarok, VLC media player, GIMP, Inkscape, Eclipse, Adobe Photoshop Lightroom, а также для различных встроенных систем, например для управления станками и бытовой техникой [3].

Для разработки было использовано Qt Creator (ранее известная под кодовым названием Greenhouse) — свободная IDE для разработки на C, C++, JavaScript и QML. Разработана Trolltech (Digia) для работы с фреймворком Qt.

Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML. Поддерживаемые компиляторы: GCC, Clang, MinGW, MSVC, Linux ICC, GCCE, RVCT, WINSCW [4].

В качестве пояснения к проделанной работе был записан видеоролик. Для захвата экрана был использован OBS. OBS (Open Broadcaster Software) — это бесплатное и открытое программное обеспечение для записи видео и трансляции в реальном времени. Оно позволяет транслировать игры, вебинары, стримы и другой контент. Пользователи могут записывать свой экран, вебкамеру, игры и другие источники видео. OBS поддерживает все основные платформы, включая Windows, macOS и Linux. Для монтажа видеоролика был использован CapCut — бесплатный видеоредактор с большим разнообразием фильтров, переходов, стикеров, шрифтов и наложений.

3 РАЗРАБОТКА ПРОГРАММЫ

Анализ данной задачи позволит нам оценить эффективность и корректность реализации алгоритмов работы с графами, а также выявить возможные улучшения и оптимизации. В ходе анализа мы рассмотрим основные этапы работы с АРМ.

Анализ кода

1) Класс House представляет собой модель дома. В классе определены различные атрибуты, такие как количество этажей, площади внутренних и внешних помещений, виды фундамента, обшивки, электрики, сантехники и теплого пола. Также есть метод `assemble_the_house`, который принимает вектор индексов и текущую площадь и устанавливает значения этих атрибутов в соответствии с выбранными характеристиками. Класс использует ряд полей для определения стоимости различных компонентов дома, включая внешнюю и внутреннюю площадь, количество свай фундамента, типы обшивки, электрики, сантехники и системы отопления. Стоимость доставки также учитывается в зависимости от общей площади дома.

2) Класс Prices, содержит информацию о ценах на различные услуги и материалы для строительства дома. Класс включает векторы с ценами за этажность, фундамент, обшивку, электрику, сантехнику, теплый пол и доставку домокомплекта. Также есть метод `change_current_prices`, который позволяет обновить эти цены, прочитав их из файла. При создании объекта класса считываются текущие цены из файла.

3) Интерфейс приложения написан с использованием библиотеки Qt.

4) Класс `change_the_directory`, наследуется от класса `QDialog`. Класс `QDialog` используется в Qt для создания модальных диалоговых окон, которые блокируют доступ к родительскому окну до тех пор, пока они не будут закрыты. Класс `change_the_directory` создает окно, в котором пользователь может обновить цены для расчета стоимости строительства дома.

5) Класс `Main_Window_1`, наследуется от класса `QMainWindow` и реализует главное окно приложения. Класс `Main_Window_1` содержит следующие элементы: 1. `Prices current_prices`; - переменная типа `Prices`, которая хранит текущие цены на товары или услуги. 2. `House house_2`; - переменная типа `House`, которая представляет дом, который собирается построить пользователь. 3. Методы для обработки событий, таких как нажатие кнопок и открытие окон. 4. Четыре слота (`on_calculate_but_clicked()`, `on_change_prices_but_clicked()`, `on_save_data_but_clicked()`, `on_pushButton_clear_clicked()`) для обработки событий. 5. Три сигнала (`signal_1()`, `signal_3()`), которые могут быть использованы для передачи данных между различными частями программы. 6. Один публичный слот (`slot_2()`), который может быть вызван другими частями программы. 7. Два приватных объекта (`window_3`, `window_4`), которые представляют окна для изменения цен и сохранения данных соответственно. 8. Приватный объект `ui`, который содержит информацию о пользовательском интерфейсе главного окна. Основная функциональность класса `Main_Window_1` заключается в обработке событий, связанных с GUI, и взаимодействии с другими частями программы через сигналы и слоты.

6) Класс `save_data_window`, наследуется от класса `QDialog`. Этот класс представляет собой диалоговое окно для сохранения данных. Класс `save_data_window` содержит следующие элементы: 1. Методы: - `explicit save_data_window(QWidget * parent = nullptr)`: конструктор класса, который принимает родительский виджет (может быть `nullptr`). `~save_data_window()`: деструктор класса. `void slot_3(Prices& catalog_3, House& house_3)`: слот, который вызывается при выполнении некоторого сигнала. Принимает два параметра: `catalog_3` типа `Prices` и `house_3` типа `House`. `void on_pushButton_cancel_clicked()`: обработчик события клика по кнопке "Отмена". - `void on_pushButton_save_clicked()`: обработчик события клика по кнопке "Сохранить". 2. Слоты (Slots): `on_pushButton_cancel_clicked()`: когда пользователь нажимает кнопку "Отмена", этот метод вызывается и выполняет

необходимые действия. `on_pushButton_save_clicked()`: когда пользователь нажимает кнопку "Сохранить", этот метод вызывается и выполняет необходимые действия. 3. Переменные: `Ui::save_data_window * ui`: указывает на экземпляр класса `save_data_window`, который содержит пользовательский интерфейс диалогового окна.

7) Главная функция `main` выполняет следующие операции: 1. Включает необходимые заголовки для использования библиотеки Qt и класса `Main_Window_1`. 2. Создает экземпляр объекта `QApplication` и передает ему аргументы командной строки. 3. Пытается открыть файл стилей `styles.css`, который находится в директории приложения, и установить его в качестве CSS-стилистики приложения. Если файл не найден, выводится предупреждение. 4. Создает экземпляр окна `Main_Window_1` и отображает его. 5. Запускает цикл обработки событий главного потока события (`a.exec()`), что позволяет приложению работать до тех пор, пока пользователь не закроет главное окно.

8) Пользовательский интерфейс выполнен в файлах с расширением `ui`.

Для данной лабораторной работы требуется `uml` диаграмма, которая позволит наглядно визуализировать методы и поля классов, а также взаимодействие классов между собой.

Класс `House` (Приложение А.1) реализует собой модель дома, где хранятся выбранные пользователем элементы строительства дома.

Класс `Price` (Приложение А.1) реализует каталог цен, с помощью которого подсчитывается финальная стоимость строительства дома.

Класс `Main_Window_1` (Приложение А.2) реализует главное окно, с помощью которого осуществляется взаимодействие с пользовательским интерфейсом.

Классы `change_the_directory` и `save_data_window` (Приложение А.3) осуществляют изменение цен в каталоге и сохранение данных соответственно.

Листинг программы указан в Приложении Б. Стили для окон приложения и каталог цен находятся в файлах css и txt соответственно (Приложении Б.11 и Приложении Б.12)

4 ТЕСТИРОВАНИЕ ПРОГРАММЫ

Для тестирования написанной программы необходимо выполнить следующие шаги:

- 1) Запустить программу.
- 2) Провести необходимые операции с калькулятором.

Для проверки достоверности калькулятора я сверилась с официальными расчетами. В ходе сверки оказалось, что калькулятор работает корректно.

Пояснение пользовательского интерфейса (Приложение В):

- С помощью радиокнопок пользователь выбирает необходимые характеристики дома (Приложение В.1).
- Кнопка «Рассчитать» рассчитывает стоимость дома (Приложение В.2).
- Кнопка «Сохранить данные» сохраняет полученные расчёты в файле txt в месте, которое указывает пользователь (Приложение В.3).
- Кнопка «Изменить цены» позволяет изменить текущие цены в каталоге (Приложение В.4).
- Кнопка «Очистить» удаляет текущие расчёты.
- Кнопка «Выход» позволяет завершить работу с приложением.

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были рассмотрены основные операции для расчета стоимости строительства дома.

В ходе работы я применила знания о работе с классами и интерфейсами и разработала алгоритм расчета стоимости дома, а также научилась использовать его с в приложении. По ходу работы был разработан калькулятор, операции с которым выполняются посредством работы с интерфейсом, разработанным с помощью средств Qt.

Были реализованы классы House и Prices, где хранятся выбранные значения для расчета и актуальные цены, а также реализую методы расчета строительства дома. По итогу работы было разработан калькулятор, с меню, которое позволяет управлять им.

В дальнейшем, полученные навыки могут быть применены для усовершенствования разработанного приложения.

В GitHub представлен полный код программы (Приложение Г.1).

Ссылка: https://github.com/SonyAkb/Laboratory-works-for-the-2-semester/tree/main/creative%20work/creative_work

На YouTube представлено видео, которое поясняет функционал программы.

Ссылка: https://youtu.be/vfr6gA_pARs

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Строительство каркасных домов в Пермском крае // Finstroy Пермь URL: <https://quizdom.ru/> (дата обращения: 12.05.24).
- 2) Герберт, Шилдт С++. Базовый курс / Шилдт Герберт. - М.: Диалектика / Вильямс, 2022. - 564 с.
- 3) Qt // Википедия URL: <https://ru.wikipedia.org/wiki/Qt> (дата обращения: 12.05.24).
- 4) Qt Creator // Википедия URL: https://ru.wikipedia.org/wiki/Qt_Creator (дата обращения: 12.05.24).

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

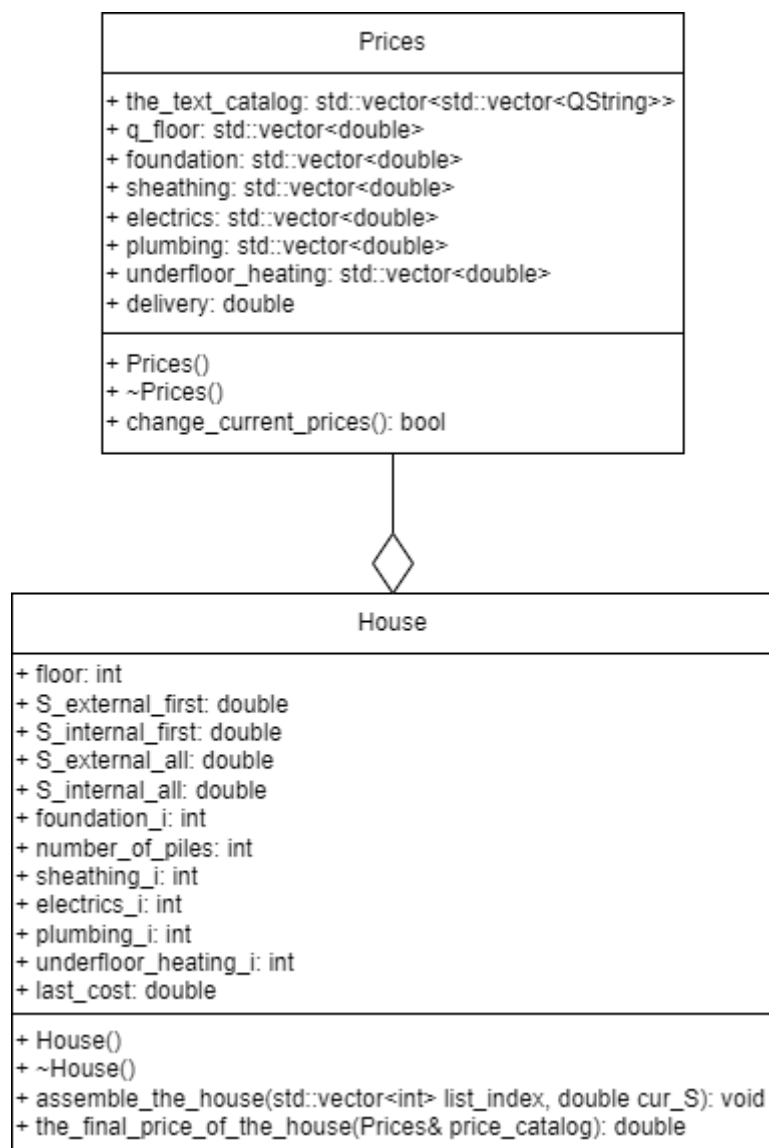


Рисунок А.1 – uml диаграмма классов House и Prices

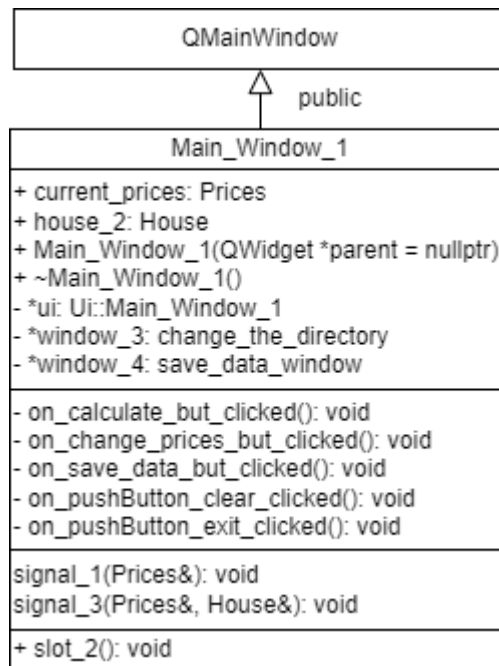


Рисунок А.2 – uml диаграмма класса Main_Window_1

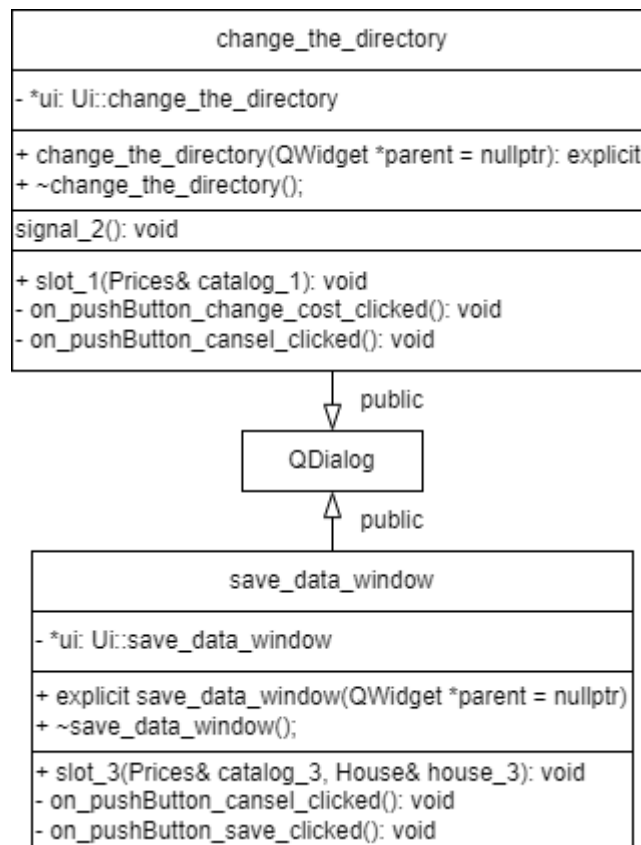


Рисунок А.3 – uml диаграммы классов change_the_directory и save_data_window

ПРИЛОЖЕНИЕ Б

Листинг Б.1 – Файл house.h

```
#ifndef HOUSE_H
#define HOUSE_H
#include "prices.h"
#include <cmath>
class House { //класс дом
public:
    int floor; //этаж: 0 - первый, 1 - первый + мансарда, 2 - второй
    double S_external_first; //площадь внешняя 1 этаж
    double S_internal_first; //площадь внутренняя 1 этаж
    double S_external_all = 0; //площадь внешняя ВСЯ
    double S_internal_all = 0; //площадь внутренняя ВСЯ
    int foundation_i; //вид фундамента - индекс
    int number_of_piles; //количество свай
    int sheathing_i; //вид обшивки - индекс
    int electrics_i; //электрика нужна или нет - индекс
    int plumbing_i; //сантехника нужна или нет - индекс
    int underfloor_heating_i; //теплый пол нужен или нет - индекс
    double last_cost = 0; //цена дома

    House() {
        floor = 0; //этаж: 0 - первый, 1 - первый + мансарда, 2 - второй
        S_external_first = 0; //площадь внешняя 1 этаж
        S_internal_first = 0; //площадь внутренняя ВСЯ
        foundation_i = 0; //вид фундамента - индекс
        sheathing_i = 0; //вид обшивки - индекс
        electrics_i = 0; //электрика нужна или нет - индекс
        plumbing_i = 0; //сантехника нужна или нет - индекс
        underfloor_heating_i = 0; //теплый пол нужен или нет - индекс
    }
    ~House() {} ;
    void assemble_the_house(std::vector<int> list_index, double
cur_S) { //характеристики дома
        floor = list_index[0]; //этажность
        if (list_index[1] == 0) {
            S_external_first = cur_S; //внешняя площадь
            S_internal_first = S_external_first * 0.87; //внутренняя площадь -
меньше внешней на 13%
        }
        else {
            S_internal_first = cur_S; //внутренняя площадь - меньше внешней на
13%
            S_external_first = (S_internal_first / 87) * 100; //внешняя
площадь
        }
        if (list_index[0] == 0) { //1 этаж
            S_external_all = S_external_first; //снаружи вся S
            S_internal_all = S_internal_first; //внутри вся S
        }
        else if (list_index[0] == 1) { //1 этаж + мансарда
            double approximate_width = sqrt(S_external_first); //примерная
длина по квадрату
            S_external_all = S_external_first + ((approximate_width - 2) *
approximate_width); //снаружи вся S
            S_internal_all = S_internal_all * 0.87; //внутри вся S
        }
        else { //2 этажа
            S_external_all = S_external_first * 2; //снаружи вся S
            S_internal_all = S_internal_first * 2; //внутри вся S
        }
    }
};
#endif
```

```

        foundation_i = list_index[2]; //вид фундамента
        number_of_piles = std::ceil(S_external_first / 3); //количество свай
        sheathing_i = list_index[3]; //вид обшивки
        electrics_i = list_index[4]; //вид электрики
        plumbing_i = list_index[5]; //вид сантехники
        underfloor_heating_i = list_index[6]; //вид
    }

    double the_final_price_of_the_house(Prices& price_catalog) { //цена дома
итоговая
        double the_entire_amount = 0; //цена дома
        if(S_external_all > 0) { //если площадь дома адекватная
            double cost_floor = S_external_all *
price_catalog.q_floor[floor]; //домокомплект
            the_entire_amount += cost_floor;
            double cost_foundation = number_of_piles *
price_catalog.foundation[foundation_i]; //фундамент
            the_entire_amount += cost_foundation;
            double cost_sheathing = S_internal_all *
price_catalog.sheathing[sheathing_i]; //обшивка
            the_entire_amount += cost_sheathing;
            double cost_electrics = S_internal_all *
price_catalog.electrics[electrics_i]; //электрика
            the_entire_amount += cost_electrics;
            double cost_plumbing = S_internal_first *
price_catalog.plumbing[plumbing_i]; //сантехника
            the_entire_amount += cost_plumbing;
            double cost_underfloor_heating = S_internal_first *
price_catalog.underfloor_heating[underfloor_heating_i]; //теплый пол
            the_entire_amount += cost_underfloor_heating;
            double cost_delivery;
            if(S_external_all <= 60.0) {
                cost_delivery = price_catalog.delivery;
            }
            else {
                cost_delivery = price_catalog.delivery *
std::ceil(S_external_all / 60 / 0.5) * 0.5;
            }
            the_entire_amount += cost_delivery;
            last_cost = the_entire_amount;
        }
        return the_entire_amount;
    }
};
#endif // HOUSE_H

```

Листинг Б.2 – Файл Prices.h

```

#ifndef PRICES_H
#define PRICES_H
#include <vector>
#include <QtGui>
#include <fstream>
#include <QFile>
#include <QString>
#include <QTextStream>

class Prices{
public:
    std::vector<std::vector<QString>> the_text_catalog{"1 этаж", "1 этаж +
мансарда", "2 этажа"},

    {"винтовой", "свайный"},

```

```

        {"нет", "ГКЛ",
        {"нет", "да"},
{"нет", "да"}, {"нет", "да"}}};
    std::vector<double> q_floor{43000,36000,43000}; //цены за этажность
    std::vector<double> foundation{6000,8000}; //цены за фундамент
    std::vector<double> sheathing{0, 2400,3000,5000}; //цены за обшивку
    std::vector<double> electrics = {0, 1500}; //электрика
    std::vector<double> plumbing = {0, 1100}; //сантехника
    std::vector<double> underfloor_heating = {0, 2300}; //теплый пол
    double delivery = 160000; //доставка домокомплекта

public:
    Prices() {
        change_current_prices();
    }
    ~Prices(){};
    bool change_current_prices() { //поменять цены в классе сейчас
        QFile file(QCoreApplication::applicationDirPath() +
"/additional_files/all_the_prices.txt"); //открываю файл с ценами
        if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
            return false;
        }
        QTextStream in(&file);
        q_floor = std::vector<double>{in.readLine().toDouble(),
in.readLine().toDouble(), in.readLine().toDouble()};
        foundation = std::vector<double>{in.readLine().toDouble(),
in.readLine().toDouble()};
        electrics = std::vector<double>{ 0, in.readLine().toDouble()};
        plumbing = std::vector<double>{0, in.readLine().toDouble()};
        underfloor_heating = std::vector<double>{0,
in.readLine().toDouble()};
        sheathing = std::vector<double>{0, in.readLine().toDouble(),
in.readLine().toDouble(), in.readLine().toDouble()};
        delivery = in.readLine().toDouble();
        file.close();
        return true;
    }
};
#endif // PRICES_H

```

Листинг Б.3 – Файл main_window_1.h

```

#ifndef MAIN_WINDOW_1_H
#define MAIN_WINDOW_1_H

#include <QMainWindow>
#include <QMessageBox>
#include <QtWidgets>
#include "house.h"
#include "prices.h"
#include "change_the_directory.h"
#include "save_data_window.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Main_Window_1; }
QT_END_NAMESPACE

class Main_Window_1 : public QMainWindow
{
    Q_OBJECT

public:
    Prices current_prices; //каталог цен

```

```

    House house_2; //дом, который собираю
    Main_Window_1(QWidget *parent = nullptr);
    ~Main_Window_1();

private slots:
    void on_calculate_but_clicked();
    void on_change_prices_but_clicked();
    void on_save_data_but_clicked();
    void on_pushButton_clear_clicked();
    void on_pushButton_exit_clicked();
private:
    Ui::Main_Window_1 *ui; //главное окно
    change_the_directory *window_3; //изменить цены
    save_data_window *window_4; //сохранить данные
signals:
    void signal_1(Prices&);
    void signal_3(Prices&, House&);
public slots:
    void slot_2();
};
#endif // MAIN_WINDOW_1_H

```

Листинг Б.4 – Файл main_window_1.cpp

```

#include "main_window_1.h"
#include "ui_main_window_1.h"
#include "other_functoins.h" //доп функции
#include "house.h" //класс дома
#include <string>
#include <vector>

Main_Window_1::Main_Window_1(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::Main_Window_1)
{
    ui->setupUi(this);
    ui->floor_1_rb->setChecked(true);
    ui->not_sh_rb->setChecked(true);
    ui->external_area_rb->setChecked(true);
    ui->piled_foundation_rb->setChecked(true);
    ui->no_plumbing_rb->setChecked(true);
    ui->no_electrics_rb->setChecked(true);
    ui->no_underfloor_heating_rb->setChecked(true);
    this->setWindowTitle("Калькулятор расчета стоимости строительства дома по финской технологии");

    window_3 = new change_the_directory; //изменить цены
    connect(this, &Main_Window_1::signal_1, window_3,
    &change_the_directory::slot_1);
    connect(window_3, &change_the_directory::signal_2, this,
    &Main_Window_1::slot_2);

    window_4 = new save_data_window; //сохранить данные
    connect(this, &Main_Window_1::signal_3, window_4,
    &save_data_window::slot_3);
}

Main_Window_1::~Main_Window_1()
{
    delete ui;
}

void Main_Window_1::slot_2() {

```

```

        current_prices.change_current_prices();
    }

void Main_Window_1::on_calculate_but_clicked() {
    if(can_convert_to_string_double(ui->lineEdit_enter_area->text()) &&
    string_to_double(ui->lineEdit_enter_area->text()) > 0){
        std::vector<std::vector<bool>> bool_vector_all_data{{ui->floor_1_rb-
>isChecked(), ui->floor_1_attic_rb->isChecked(), ui->floor_2_rb-
>isChecked()},//этaж
                                                    {ui-
>external_area_rb->isChecked(), ui->internal_area->isChecked()},//площадь
                                                    {ui-
>screw_foundation_rb->isChecked(), ui->piled_foundation_rb-
>isChecked()},//фундамент
                                                    {ui->not_sh_rb-
>isChecked(), ui->gkl_rb->isChecked(), ui->board_walls->isChecked(), ui-
>board_walls_ceiling->isChecked()},//обшивкa
                                                    {ui-
>no_electrics_rb->isChecked(), ui->yes_electrics_rb->isChecked()},//электрикa
                                                    {ui-
>no_plumbing_rb->isChecked(), ui->yes_plumbing_rb->isChecked()},//сантехникa
                                                    {ui-
>no_underfloor_heating_rb->isChecked(), ui->yes_underfloor_heating_rb-
>isChecked()}}; //теплый пол

        std::vector<int> list_index = all_list_index(bool_vector_all_data);
        house_2.assemble_the_house(list_index, string_to_double(ui-
>lineEdit_enter_area->text())); //запрос на дом
        double final_cost =
house_2.the_final_price_of_the_house(current_prices); //итоговая цена
        ui->the_final_prices->setText(double_to_string(final_cost));
    }
    else{//введен не текст вместо S
        QMessageBox message_Box_1;
        message_Box_1.setText("Ошибка");
        message_Box_1.setInformativeText("Некорректная площадь");
        message_Box_1.setWindowTitle("Ошибка");
        message_Box_1.setIcon(QMessageBox::Critical);
        QFont font("Helvetica", 16);
        message_Box_1.setFont(font);
        message_Box_1.setInformativeText("Некорректная площадь");
        message_Box_1.setWindowTitle("Ошибка");
        message_Box_1.exec();// Показываем сообщение об ошибке
    }
}

void Main_Window_1::on_change_prices_but_clicked() {
    window_3->show();
    emit signal_1(current_prices);
}

void Main_Window_1::on_save_data_but_clicked()
{
    if(string_to_double(ui->lineEdit_enter_area->text()) > 0){
        window_4->show();
        emit signal_3(current_prices, house_2);
    }
    else{
        QMessageBox message_Box_1;
        message_Box_1.setText("Ошибка");
        message_Box_1.setInformativeText("Некорректная площадь");
        message_Box_1.setWindowTitle("Ошибка");
        message_Box_1.setIcon(QMessageBox::Critical);
        QFont font("Helvetica", 16);
    }
}

```

```

        message_Box_1.setFont(font);
        message_Box_1.setInformativeText("Некорректная площадь");
        message_Box_1.setWindowTitle("Ошибка");
        message_Box_1.exec(); // Показываем сообщение об ошибке
    }
}

void Main_Window_1::on_pushButton_clear_clicked()
{
    ui->lineEdit_enter_area->setText("");
    ui->the_final_prices->setText("0");
}

void Main_Window_1::on_pushButton_exit_clicked()
{
    this->close();
}

```

Листинг Б.5 – Файл change_the_directory.h

```

#ifndef CHANGE_THE_DIRECTORY_H
#define CHANGE_THE_DIRECTORY_H

#include <QDialog>
#include <QWidget>
#include "prices.h"
namespace Ui {
class change_the_directory;
}

class change_the_directory : public QDialog
{
    Q_OBJECT

public:
    explicit change_the_directory(QWidget *parent = nullptr);
    ~change_the_directory();
private:
    Ui::change_the_directory *ui;
public slots:
    void slot_1(Prices& catalog_1);
private slots:
    void on_pushButton_change_cost_clicked();
    void on_pushButton_cancel_clicked();
signals:
    void signal_2();
};

#endif // CHANGE_THE_DIRECTORY_H

```

Листинг Б.6 – Файл change_the_directory.cpp

```

#include "change_the_directory.h"
#include "ui_change_the_directory.h"
#include <iomanip>
#include <QString>
#include <QFile>
#include <QLineEdit>

change_the_directory::change_the_directory(QWidget *parent) :
    QDialog(parent),

```

```

        ui(new Ui::change_the_directory)
    {
        ui->setupUi(this);
        this->setWindowTitle("Изменение цен");
    }

change_the_directory::~change_the_directory()
{
    delete ui;
}

void change_the_directory::slot_1(Prices &catalog_1)
{
    ui->lineEdit_1_floor->setText(QString::number(catalog_1.q_floor[0], 'f',
2)); //1 этаж
    ui->lineEdit_1_floor_attic->setText(QString::number(catalog_1.q_floor[1],
'f', 2)); //1 этаж + веранда
    ui->lineEdit_2_floor->setText(QString::number(catalog_1.q_floor[2], 'f',
2)); //2 этажа
    ui->lineEdit_pile->setText(QString::number(catalog_1.foundation[1], 'f',
2)); //фундамент свайный
    ui->lineEdit_screw->setText(QString::number(catalog_1.foundation[0], 'f',
2)); //фундамент винтовой
    ui->lineEdit_gkl->setText(QString::number(catalog_1.sheathing[1], 'f',
2)); //обшивка ГКЛ
    ui->lineEdit_board_well->setText(QString::number(catalog_1.sheathing[2],
'f', 2)); //обшивка вагонка стены
    ui->lineEdit_board_well_celing-
>setText(QString::number(catalog_1.sheathing[3], 'f', 2)); //обшивка вагонка
стены + потолок
    ui->lineEdit_plumbing->setText(QString::number(catalog_1.plumbing[1],
'f', 2)); //сантехника
    ui->lineEdit_electrics->setText(QString::number(catalog_1.electrics[1],
'f', 2)); //электрика
    ui->lineEdit_floor_hot-
>setText(QString::number(catalog_1.underfloor_heating[1], 'f', 2)); //теплый
пол
    ui->lineEdit_delivery->setText(QString::number(catalog_1.delivery, 'f',
2)); //доставка
}

void change_the_directory::on_pushButton_change_cost_clicked(/*Prices&
catalog_1*/)
{
    QFile file_1(QCoreApplication::applicationDirPath() +
"/additional_files/all_the_prices.txt"); //открываю файл с ценами
    if (!file_1.open(QIODevice::ReadOnly | QIODevice::Text)) {
        return;
    }
    QTextStream in_1(&file_1); //чтение каталога цен
    QFile file_2(QCoreApplication::applicationDirPath() +
"/additional_files/temporary_file.txt");
    if (file_2.open(QIODevice::WriteOnly)) {
        QString the_value_in_question;
        QString line;
        foreach (const QString &name, QStringList({"lineEdit_1_floor",
"lineEdit_1_floor_attic", "lineEdit_2_floor",
"lineEdit_screw", "lineEdit_pile", "lineEdit_electrics",
"lineEdit_plumbing", "lineEdit_floor_hot",
"lineEdit_gkl", "lineEdit_board_well", "lineEdit_board_well_celing",
"lineEdit_delivery"})) { // Список имен текстовых полей

```



```

        QLineEdit * lineEdit = this->findChild<QLineEdit * >(name); //
Находим текстовое поле с данным именем
        the_value_in_question = lineEdit->text();
        line = in_1.readLine();

        if (lineEdit != nullptr) { // Проверяем, было ли найдено
текстовое поле
            if(the_value_in_question.toDouble() > 0){//1 этаж

file_2.write(QString::number(the_value_in_question.toDouble(), 'f',
2).toStdString().c_str());
                file_2.write("\n");
            }
            else{
                file_2.write(line.toStdString().c_str());
                file_2.write("\n");
            }
        }
        file_2.close();//закрываю
        file_1.close();//закрываю
        file_1.open(QIODevice::WriteOnly);
        file_2.open(QIODevice::ReadOnly | QIODevice::Text);

        QByteArray data;
        while (!file_2.atEnd()) {
            data += file_2.read(1024); // Читаем данные блоками по 1 КБ
        }
        file_1.write(data); // Записываем данные в новый файл
        file_1.close();//закрываю
        file_2.close();//закрываю
        file_2.remove();//удаляю временный файл
        emit signal_2();
    }// не удалось создать/открыть файл
    else{
        file_1.close();
    }
    this->close();
}

void change_the_directory::on_pushButton_cansel_clicked()
{
    this->close();
}

```

Листинг Б.7 – Файл save_data_window.h

```

#ifndef SAVE_DATA_WINDOW_H
#define SAVE_DATA_WINDOW_H

#include <QDialog>
#include <QFileDialog>
#include "house.h"
#include "prices.h"

namespace Ui {
class save_data_window;
}

class save_data_window : public QDialog
{
    Q_OBJECT

public:
    explicit save_data_window(QWidget *parent = nullptr);

```

```

        ~save_data_window();
private:
    Ui::save_data_window *ui;
public slots:
    void slot_3(Prices& catalog_3, House& house_3);
private slots:
    void on_pushButton_cancel_clicked();
    void on_pushButton_save_clicked();
};

#endif // SAVE_DATA_WINDOW_H

```

Листинг Б.8 – Файл save_data_window.cpp

```

#include "save_data_window.h"
#include "ui_save_data_window.h"

save_data_window::save_data_window(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::save_data_window)
{
    ui->setupUi(this);
    this->setWindowTitle("Сохранить...");
}

save_data_window::~save_data_window()
{
    delete ui;
}

void save_data_window::slot_3(Prices &catalog_3, House &house_3)
{
    int i = 0;
    ui->label_amount_floor_2->
>setText(catalog_3.the_text_catalog[i++][house_3.floor]); //этажность
    ui->label_S_out_1f_2->setText(QString::number(house_3.S_external_first,
'f', 2)); //внешняя 1 этаж
    ui->label_S_in_1f_2->setText(QString::number(house_3.S_internal_first,
'f', 2)); //внутренняя 1 этаж
    ui->label_S_in_all_2->setText(QString::number(house_3.S_internal_all,
'f', 2)); //внутренняя вся
    ui->label_found_2->
>setText(catalog_3.the_text_catalog[i++][house_3.foundation_i]); //фундамент
    ui->label_sheathing_2->
>setText(catalog_3.the_text_catalog[i++][house_3.sheathing_i]); //обшивка
    ui->label_plumbing_2->
>setText(catalog_3.the_text_catalog[i++][house_3.plumbing_i]); //сантехника
    ui->label_electrics_2->
>setText(catalog_3.the_text_catalog[i++][house_3.electrics_i]); //электрика
    ui->label_hot_floor_2->
>setText(catalog_3.the_text_catalog[i++][house_3.underfloor_heating_i]); //теп
лый пол
    ui->label_result_2->setText(QString::number(house_3.last_cost, 'f',
2)); //итог
}

void save_data_window::on_pushButton_cancel_clicked()
{
    this->close();
}

void save_data_window::on_pushButton_save_clicked()
{

```

```

        const QString defaultPath =
QStandardPaths::standardLocations(QStandardPaths::DesktopLocation).first();
        // Создание объекта QFileDialog для выбора пути к файлу
        QString file_Path = QFileDialog::getSaveFileName(nullptr,
                                                         "Сохранить файл",
                                                         defaultPath +
                                                         ".txt ( * .txt * )");

        if (file_Path.isEmpty()) { // Пользователь отменил сохранение
            return;
        }
        if (!(file_Path[file_Path.length() - 1] == 't' &&
file_Path[file_Path.length() - 2] == 'x' &&
        file_Path[file_Path.length() - 3] == 't' &&
file_Path[file_Path.length() - 4] == '.')){
            file_Path += ".txt";
        }

        QFile file_1(file_Path); //открываю файл с расширением txt
        if (file_1.open(QIODevice::WriteOnly | QIODevice::Append)){
            file_1.write((QDateTime::currentDateTime().toString("yyyy-MM-dd_hh-
mm-ss")).toStdString() + "\n").c_str());
            file_1.write((ui->label_name->text().toStdString() + " " + ui-
>lineEdit_name->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_phone->text().toStdString() + " " + ui-
>lineEdit_phone->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_found_1->text().toStdString() + ": " + ui-
>label_found_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_S_out_1f_1->text().toStdString() + ": " + ui-
>label_S_out_1f_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_S_in_1f_1->text().toStdString() + ": " + ui-
>label_S_in_1f_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_S_in_all_1->text().toStdString() + ": " + ui-
>label_S_in_all_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_found_1->text().toStdString() + ": " + ui-
>label_found_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_sheathing_1->text().toStdString() + ": " +
ui->label_sheathing_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_plumbing_1->text().toStdString() + ": " + ui-
>label_plumbing_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_electrics_1->text().toStdString() + ": " +
ui->label_electrics_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_hot_floor_1->text().toStdString() + ": " +
ui->label_hot_floor_2->text().toStdString() + "\n").c_str());
            file_1.write((ui->label_result_1->text().toStdString() + ": " + ui-
>label_result_2->text().toStdString() + "\n\n").c_str());
        }
        this->close();
    }
}

```

Листинг Б.9 – Файл other_fuctions.cpp

```

#ifndef OTHER_FUNCIONS_H
#define OTHER_FUNCIONS_H

#include <sstream>
#include <QString>
#include <iostream>
#include <iomanip>
#include <vector>

bool can_convert_to_string_double(const QString &string) {
    if (string == '0'){
        return 1;
    }
}

```

```

        return string.toDouble() != 0; // Проверяем, что преобразование прошло
        успешно
    }

double string_to_double(const QString &str) { // Функция для перевода из
строки в double
    return str.toDouble();
}

QString double_to_string(double value) {
    std::ostringstream oss; // Создаем временный поток для записи чисел в
строку с фиксированной точностью
    oss << std::fixed << std::setprecision(2) << value; // Устанавливаем
точность (количество знаков после запятой) равным двум
    return QString::fromStdString(oss.str()); // Возвращаем QString из
полученной строки
}

int getting_data(std::vector<bool> bool_vector) { // получаю индекс выбранного
варианта
    int size_vector = static_cast<unsigned int>(bool_vector.size()); //
Приводим размер вектора к беззнаковому типу
    for(int i=0; i < size_vector ; i++){
        if (bool_vector[i]){
            return i;
        }
    }
    return 0;
}

std::vector<int> all_list_index(std::vector<std::vector<bool>>
all_bool_vect) { // вектор избранных
    std::vector<int> list_index; // сам вектор
    int size_vector = static_cast<unsigned int>(all_bool_vect.size()); //
Приводим размер вектора к беззнаковому типу
    for(int i=0; i < size_vector ; i++){
        list_index.push_back(getting_data(all_bool_vect[i]));
    }
    return list_index;
}

#endif // OTHER_FUNCIONS_H

```

Листинг Б.10 – Файл main.cpp

```

#include "main_window_1.h"
#include <QApplication>
#include <QTextStream>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QFile file(QCoreApplication::applicationDirPath() +
"/additional_files/styles.css");
    if (file.exists()) {
        file.open(QFile::ReadOnly | QFile::Text);
        QTextStream stream(&file);
        a.setStyleSheet(stream.readAll());
        file.close();
    } else {
        qWarning() << "Unable to open file styles.qss";
    }

    Main_Window_1 w;
}

```

```
w.show();  
  
return a.exec();  
}
```

Листинг Б.11 – Файл styles.css

```
QWidget {  
    background-color: #DCDDCE; }  
  
QPushButton {  
    color: #2C3531;  
    background-color: #88B5A2;  
}  
  
QPushButton:hover {  
    background-color: #B1D9F2;  
}  
  
QPushButton:pressed {  
    background-color: #accfc0;  
    padding: 4px;  
}  
  
QFont {  
    color: #164147;  
}  
  
QLabel {  
    color: #164147;  
}  
  
QLineEdit, QTextEdit {  
    color: #164147;  
    background-color: #c7d6d0;  
}  
  
QRadioButton {  
    color: #164147;  
}  
  
QRadioButton::indicator {  
    width: 10px;  
    height: 10px;
```

```
border-radius: 7px;
}
QRadioButton::indicator:checked {
    background-color: #418393;
    border: 2px solid #88B5A2;
}
QRadioButton::indicator:unchecked {
    background-color: #B1D9F2;
    border: 2px solid #88B5A2;
}
```

Листинг Б.12 – Файл all_the_prices.txt

```
43000.00
36000.00
43000.00
6000.00
8000.00
1500.00
1100.00
2300.00
3120.00
3000.00
5000.00
160000.00
```

ПРИЛОЖЕНИЕ В

Калькулятор расчета стоимости строительства дома по финской технологии

Этажность дома

- ☒ 1 этаж
- ☐ 1 этаж + мансарда
- ☐ 2 этажа

Площадь дома

- ☒ Внешняя
- ☐ Внутренняя

Введите площадь 1 этажа

кв. м

Обшивка

- ☒ не нужна
- ☐ ГКЛ
- ☐ вагонка: стены
- ☐ вагонка: стены + потолок

Фундамент

- ☐ Винтовой
- ☒ Свайный

Электрика

- ☒ не нужна
- ☐ нужна

Теплый пол

- ☒ не нужен
- ☐ нужен

Сантехника

- ☒ не нужна
- ☐ нужна

0 ₪

Рисунок В.1 – пользовательский интерфейс

Калькулятор расчета стоимости строительства дома по финской технологии

Этажность дома	Площадь дома	Введите площадь 1 этажа	
<input type="radio"/> 1 этаж	<input checked="" type="radio"/> Внешняя	<input type="text" value="45"/>	кв. м
<input checked="" type="radio"/> 1 этаж + мансарда	<input type="radio"/> Внутренняя	<input type="button" value="Очистить"/>	
<input type="radio"/> 2 этажа		Фундамент	Электрика
		<input type="radio"/> Винтовой	<input type="radio"/> не нужна
Обшивка		<input checked="" type="radio"/> Свайный	<input checked="" type="radio"/> нужна
<input type="radio"/> не нужна		Теплый пол	Сантехника
<input checked="" type="radio"/> ГКЛ		<input type="radio"/> не нужен	<input checked="" type="radio"/> не нужна
<input type="radio"/> вагонка: стены		<input checked="" type="radio"/> нужен	<input type="radio"/> нужна
<input type="radio"/> вагонка: стены + потолок			
3207054.32 ₽			
<input type="button" value="Сохранить данные"/>	<input type="button" value="Расчитать"/>	<input type="button" value="Изменить цены"/>	
			<input type="button" value="Выход"/>

Рисунок В.2 – расчёт стоимости строительства дома

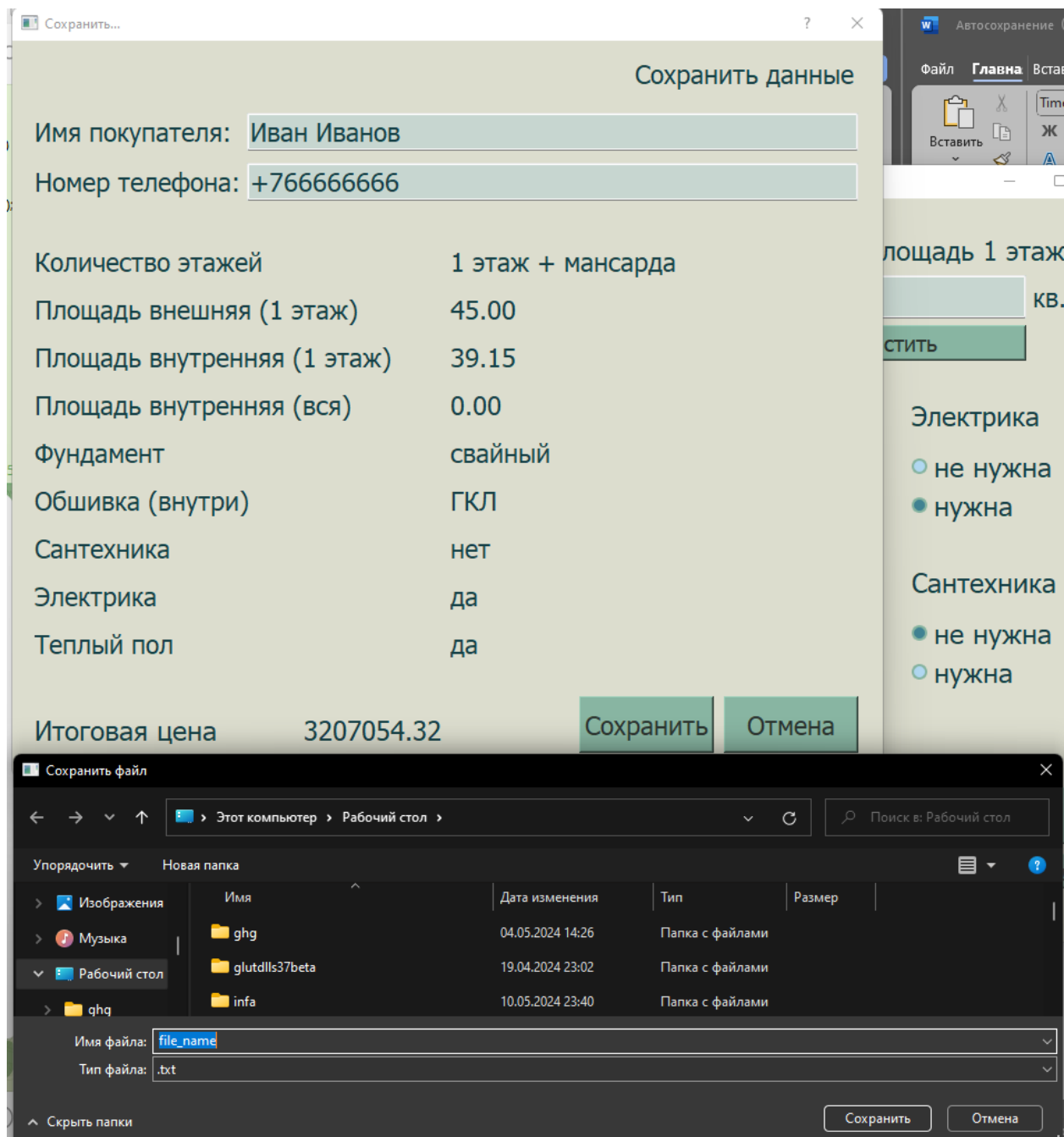


Рисунок В.3 – сохранение данных

Изменение цен

Изменение цен в каталоге

Стоимость домокомплекта за 1 кв. м

1 этаж	43000.00	1 этаж + мансарда	36000.00	2 этажа	43000.00
--------	----------	-------------------	----------	---------	----------

Стоимость фундамента (за одну сваю)

Свайный фундамент	8000.00	Винтовой фундамент	6000.00
-------------------	---------	--------------------	---------

Стоимость обшивки дома (внутри) за 1 кв. м

ГКЛ	3120.00	Вагонка (стены)	3000.00	Вагонка (стены + потолок)	5000.00
-----	---------	-----------------	---------	---------------------------	---------

Стоимость за 1 кв. м

Сантехника	1100.00	Электрика	1500.00
Теплый пол	2300.00		

Стоимость доставки за 1 машину

160000.00

Изменить цены Отмена

Рисунок В.4 – изменение цен

ПРИЛОЖЕНИЕ Г

[Laboratory-works-for-the-2-semester](#) / [creative work](#) / [creative_work](#) / 

 **SonyAkb** Delete creative work/creative_work/uml others.drawio






Name
 ..
 code
 uml - house and price.drawio.png
 uml main wind.drawio.png
 uml others.drawio.png

Рисунок Г.1 – результат работы в GitHub