Введение:

**Актуальность темы**

В современном строительстве возрастает спрос на автоматизацию проектирования, особенно при работе с железобетонными конструкциями. Подпорные стены уголкового типа широко применяются в гражданском и промышленном строительстве, однако их проектирование и последующий выпуск документации - остается трудоемким процессом. Традиционные методы создания опалубочных чертежей и BIM-моделей требуют значительных временных затрат и подвержены ошибкам из-за ручной отрисовки и заполнения данных спецификаций. Автоматизация этих процессов позволит сократить сроки проектирования и минимизировать ошибки, вызванные человеческим фактором.

**Степень проработки проблемы**

На сегодняшний день существуют решения для автоматизации проектирования железобетонных конструкций, однако специализированных инструментов для подпорных стен уголкового типа недостаточно. Большинство существующих BIM-библиотек и скриптов (например, для Revit или Tekla Structures) предлагают лишь базовые элементы, не учитывающие всех особенностей железобетонных конструкций.

**Цель и задачи исследования**

Цель работы – разработка алгоритма и программного инструмента для автоматизированного создания опалубочных чертежей и BIM-моделей подпорных стен уголкового типа по заранее определенным параметрам. Эти параметры предполагается заполнять в табличном формате в программе Excel. Так же на основе данных этих таблиц в дальнейшем будут сформированы Ведомости объемов работ и все спецификации, выводимые на листы чертежей.

Задачи исследования:

1. Анализ конструктивных особенностей подпорных стен уголкового типа;
2. Исследование существующих методов автоматизации в BIM-проектировании;
3. Разработка параметрической модели подпорной стены с возможностью адаптации под различные геометрические особенности;
4. Создание алгоритма генерации опалубочных чертежей и BIM-модели;
5. Реализация скрипта (на Python без использования Grasshopper) для автоматизации процесса;
6. Тестирование разработанного решения на реальных проектах;

Цели исследования:

1. Сокращение времени проектирования;
2. Унификация процесса моделирования подпорных стен;
3. Снижение количества ошибок и несоответствий между разными разделами документации (между ведомостью объемов работ и спецификациями на листах);

Упрощение взаимодействия между конструкторами и BIM-проектировщиками.

Ключевые особенности подхода:

1. Отказ от Grasshopper в пользу прямого программирования на Python обусловлен тем, что это повышает гибкость, увеличивает скорость обработки данных и снижает зависимость от дополнительного программного обеспечения.
2. Использование Excel как интуитивного интерфейса для инженеров (ввод параметров, табличные расчеты).
3. Сквозная автоматизация (от расчетов до BIM-модели) без ручных операций.

Предпосылки (исходные данные) работы:

Существует два основных этапа проектирования сооружения: стадия Проектной документации и стадия Рабочей Документации. Чтобы понять, чем отличается рабочая документация от проектной, стоит более подробно остановиться на ключевых особенностях каждой из них.

Стадия Проект - предназначена для прохождения экспертизы и защиты проектных решений. Проект строительства объекта включает функциональные, архитектурные, технические, инженерные решения. Все они отражаются в графическом и текстовом форматах. Если говорить о том, что включает проектная документация и рабочая документация, отличия состоят в детализации. Проектная документация включает более широкий спектр информации об объекте, но прорабатывается с невысокой степенью детализации, главное она должна пройти государственную или негосударственную экспертизу на соответствие строительным нормам и обеспечению безопасности. После чего на ее основе разрабатывается рабочая документация.

Стадия разработки Рабочей документации является основной и неотъемлемой частью всей документации по строительному объекту. Содержит информацию, установленную законодательством и позволяющую строительным бригадам осуществлять производственный процесс. Комплект включает в себя рабочие чертежи и схемы, описание решений, спецификацию материалов и изделий. Существует ряд требований, которых следует придерживаться, занимаясь подготовкой рабочей документации.

Текстовая часть содержит сведения в отношении объекта капитального строительства, описание принятых технических и иных решений, пояснения, ссылки на нормативные и (или) технические документы, используемые при подготовке проектной документации и результаты расчетов, обосновывающие принятые решения. Графическая часть отображает принятые технические и иные решения и выполняется в виде чертежей, схем, планов и других документов в графической форме.

В данной работе я буду рассматривать раздел разработки Рабочей документации для комплекта железобетонных элементов КЖ-ПСТ.

**Глава 3. Сравнительный анализ методов автоматизированного проектирования подпорных стен**

**3.1. Введение и постановка задачи**

В современной практике проектирования подпорных стен особую актуальность приобретают методы автоматизированного создания чертежей и 3D-моделей. В данной главе проводится сравнительный анализ трех ключевых подходов:

1. Чистый код на Python с использованием API AutoCAD
2. Программирование на VBA во встроенной среде AutoCAD
3. Визуальное программирование в Grasshopper (с экспортом в AutoCAD)

Цель исследования - выявить оптимальный метод для различных сценариев проектирования подпорных стен на основе комплексной оценки по техническим и эргономическим критериям.

**3.2. Методология исследования**

Для объективного сравнения применялись следующие методы:

* Функциональное тестирование (создание типовых элементов подпорных стен)
* Хронометраж выполнения операций
* Анализ сложности реализации
* Оценка точности и надежности результатов
* Исследование возможностей параметризации
* Базовые параметры для сравнения:
* Время разработки решения
* Быстродействие при выполнении
* Гибкость и адаптируемость
* Требования к квалификации пользователя
* Интеграционные возможности

3.3. Сравнительный анализ методов

3.3.1. Чистый код на Python

Преимущества:

* Высокая гибкость и контроль над процессом проектирования
* Доступ к современным библиотекам (numpy для расчетов, pandas для работы с данными)
* Возможность сложной параметризации и оптимизации конструкций
* Интеграция с внешними расчетными модулями
* Поддержка Объекто-Ориентированного Программирования для создания сложных иерархических моделей
* Кроссплатформенность и перспективность развития

Недостатки:

* Требуется дополнительная настройка среды выполнения
* Более высокий порог входа для программирования
* Необходимость глубокого знания API AutoCAD
* Относительно медленное выполнение при использовании COM-интерфейса
* Оптимальная область применения:
* Комплексные проекты с нестандартными решениями
* Системы с интеграцией внешних расчетных модулей
* Долгосрочные проекты, требующие дальнейшего развития

3.3.2. Чистый код на VBA

Преимущества:

* Полная интеграция с AutoCAD без дополнительных настроек
* Высокая скорость выполнения скриптов
* Доступ ко всем функциям AutoCAD API
* Низкий порог входа для базовых задач

Недостатки:

* Устаревшая среда разработки и язык программирования
* Ограниченные возможности работы с внешними данными
* Отсутствие современных парадигм программирования
* Проблемы с поддержкой в новых версиях AutoCAD
* Сложность сопровождения сложных проектов
* Оптимальная область применения:
* Быстрое решение стандартных задач
* Простые макросы для рутинных операций
* Проекты с жесткими требованиями к скорости выполнения

3.3.3. Grasshopper + экспорт в AutoCAD

Преимущества:

* Визуальная интуитивно понятная среда разработки
* Быстрое прототипирование и итерации
* Наглядное представление логики алгоритмов
* Широкие возможности параметрического моделирования
* Большое количество специализированных плагинов

Недостатки:

* Ограниченная производительность при сложных вычислениях
* Проблемы с организацией больших проектов
* Зависимость от сторонних плагинов
* Необходимость дополнительных операций экспорта
* Ограниченные возможности отладки
* Оптимальная область применения:
* Концептуальное проектирование и исследования
* Параметрические модели средней сложности
* Быстрая визуализация идей и вариантов

**3.4. Количественное сравнение методов**

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **Python** | **VBA** | **Grasshopper** |
| **Время разработки** | Высокое | Низкое | Среднее |
| **Скорость выполнения** | Средняя | Высокая | Низкая |
| **Гибкость** | Очень высокая | Низкая | Высокая |
| **Параметризация** | Полная | Ограниченная | Полная |
| **Поддержка сообщества** | Очень большая | Малая | Большая |
| **Перспективы развития** | Отличные | Плохие | Хорошие |

**3.6. Выводы и рекомендации**

Проведенный анализ позволяет сформулировать следующие заключения:

1. Для сложных инженерных задач с требованиями к точности расчетов и интеграцией с внешними системами оптимальным выбором является Python, несмотря на более высокие начальные затраты.
2. Для рутинных операций в среде AutoCAD, особенно при работе с устаревшими версиями, VBA сохраняет свою актуальность благодаря простоте и скорости выполнения.
3. Для концептуального проектирования и параметрических исследований Grasshopper предоставляет наиболее эффективный рабочий процесс, хотя и имеет ограничения при работе с большими моделями.

Перспективы дальнейших исследований:

* Разработка гибридных подходов (например, Grasshopper + Python)
* Автоматизация проверки соответствия нормативным требования
* Интеграция с системами конечно-элементного анализа
* Оптимизация геометрии подпорных стен с использованием машинного обучения

Выбор конкретного метода должен основываться на требованиях проекта, доступных ресурсах и квалификации команды. В перспективе наблюдается четкая тенденция в сторону использования Python и визуального программирования, в то время как значение VBA постепенно снижается.

В рамках настоящей магистерской диссертации для автоматизации проектирования подпорных стен был выбран метод программирования на Python с использованием библиотеки PyAutocad и API Tekla Structures. Данный выбор обусловлен следующими ключевыми факторами:

1. Соответствие целям и задачам исследования

Проект требовал реализации сложной параметризации геометрии подпорных стен c возможностью последующего расширения функционала и созданием чертежей и моделей в двух разных САП-комплексах одновременно.

Python, в отличие от VBA и Grasshopper, предоставил полный контроль над всеми аспектами проектирования и системный подход к решению инженерных задач.

2. Технические преимущества

Выбор Python обусловлен его уникальными техническими характеристиками:

1. Возможность реализации ООП-подхода, что критически важно для создания:

* Иерархических классов элементов стены
* Системы наследования типов конструкций
* Инкапсуляции параметров класса стены (Wall)

1. Интеграция с внешними САПР через .NET API, обеспечивающая:

* Высокую производительность
* Прямой доступ к внутренним объектам AutoCAD
* Минимальные накладные расходы

В данном пункте имеется в виду, что для работы этого приложения не требуется покупать какие-то сторонние приложения, как в случае с Grasshopper, что существенно упрощает работу с ним.

1. Научная новизна и перспективность

В дальнейшем развитии приложения, Python позволяет реализовать инновационные аспекты исследования:

* Алгоритмы оптимизации формы стены на основе методов машинного обучения
* Автоматическую генерацию расчетной документации
* Параметрическую адаптацию к изменяющимся условиям

Эти возможности недостижимы при использовании VBA и требуют сложных обходных решений в Grasshopper.

Заключение:

Выбор Python в качестве основного инструмента автоматизации доказал свою эффективность на всех этапах исследования:

* На этапе разработки - благодаря богатому инструментарию и современным практикам программирования
* В процессе тестирования - за счет надежности и предсказуемости результатов
* При внедрении - благодаря гибкости и адаптируемости решения

Этот подход не только полностью удовлетворил требования текущего проекта, но и создал надежный фундамент для дальнейших исследований в области автоматизированного проектирования инженерных сооружений. Реализованное решение демонстрирует преимущества системного программирования перед традиционными методами в САПР и открывает новые возможности для цифровой трансформации в проектировании.

**Описание принципа работы:**

Необходимо сразу внести ясность – в данной программе не будут фигурировать проверки и расчеты подпорных стен, так как она создается в предпосылке работы на том этапе проектирования, когда их результаты уже получены и учтены.

Первым шагом стала разработка Excel файла – и структурирование всех необходимых исходных данных и параметризация поперечного сечения железобетонной монолитной подпорной стены уголкового типа. Для демонстрации работы программы и всех вспомогательных файлов проекта, было выбрано искусственное сооружение – подпорная стена ИССО 1.1.1, состоящая из трех отдельных секций С1.1, С1.2 и С1.3. Каждая секция имеет одинаковую конфигурацию, но различные параметры.

Параметризация позволяет быстро изменять геометрию стены под конкретные условия, оптимизируя её по несущей способности и внешнему виду.

Поперечное сечение уголковой подпорной стены состоит из двух основных элементов:

1. Вертикальной стеновой части (консоль) – воспринимает давление грунта;
2. Горизонтальной части фундамент (ростверк) – обеспечивает устойчивость и предотвращает опрокидывание.

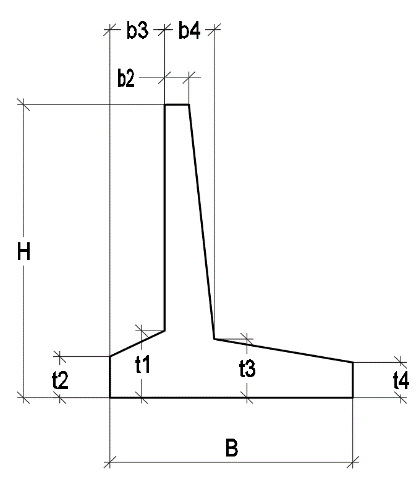


Рис. 1 – Параметризированное сечение железобетонной подпорной стены уголкового типа, где:

H // height end, height start – общая высота (различная для начала и конца подпорной стены);

B // foundation\_width – ширина фундамента подпорной стены;

b2 // top\_wall\_width – ширина стены наверху подпорной стены;

b3 // edge\_distance – расстояние от стены до границы подпорной стены;

b4 // bottom\_wall\_width – ширина стены на границе с ростверком;

t1 // t1 – толщина перекрытия 1 у стены;

t2 // t2 – толщина перекрытия 1 у насыпи;

t3 // t3 – толщина перекрытия 2 у стены;

t4 // t4 – толщина перекрытия 2 у насыпи;

В таблице Excel файла Data все параметры и необходимые характеристики выглядят вот так:

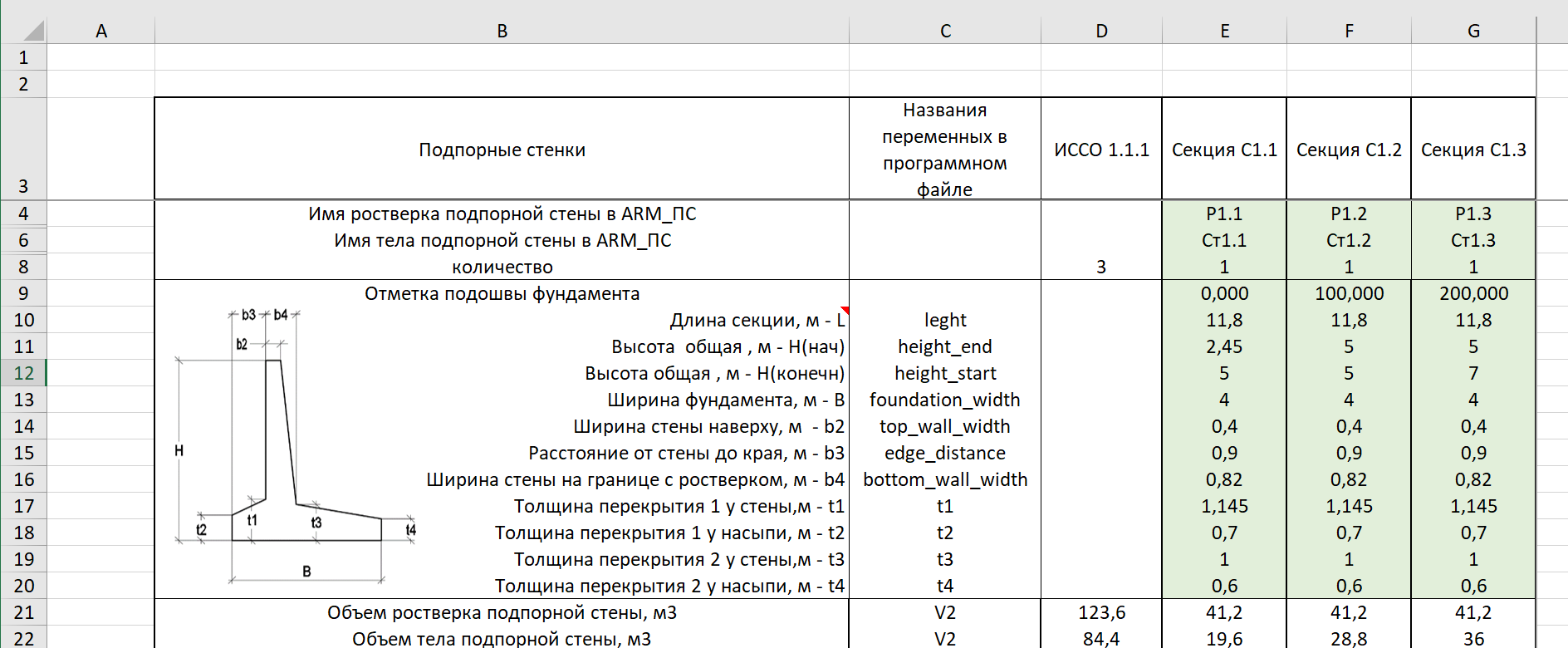


Рис.2 – Параметры подпорной стены и ее секций в файле Data

Отмеченное зеленым цветом подлежит заполнению – вручную, белым –вычисляется автоматически.

Так же на этой же вкладке файла Data производится заполнение некоторых параметров и вычисление объемов для заполнения таблиц спецификаций, выводимых моей программой на лист опалубочного, чертежа и включаемых в отдельную Ведомость объемов работ. Не все параметры из этого перечня будут напрямую задействованы в данной работе, но они необходимы для общего понимания конструкции и состава ИССО, а также заполнения Ведомостей объемов работ и возможности дальнейшего развития программы.

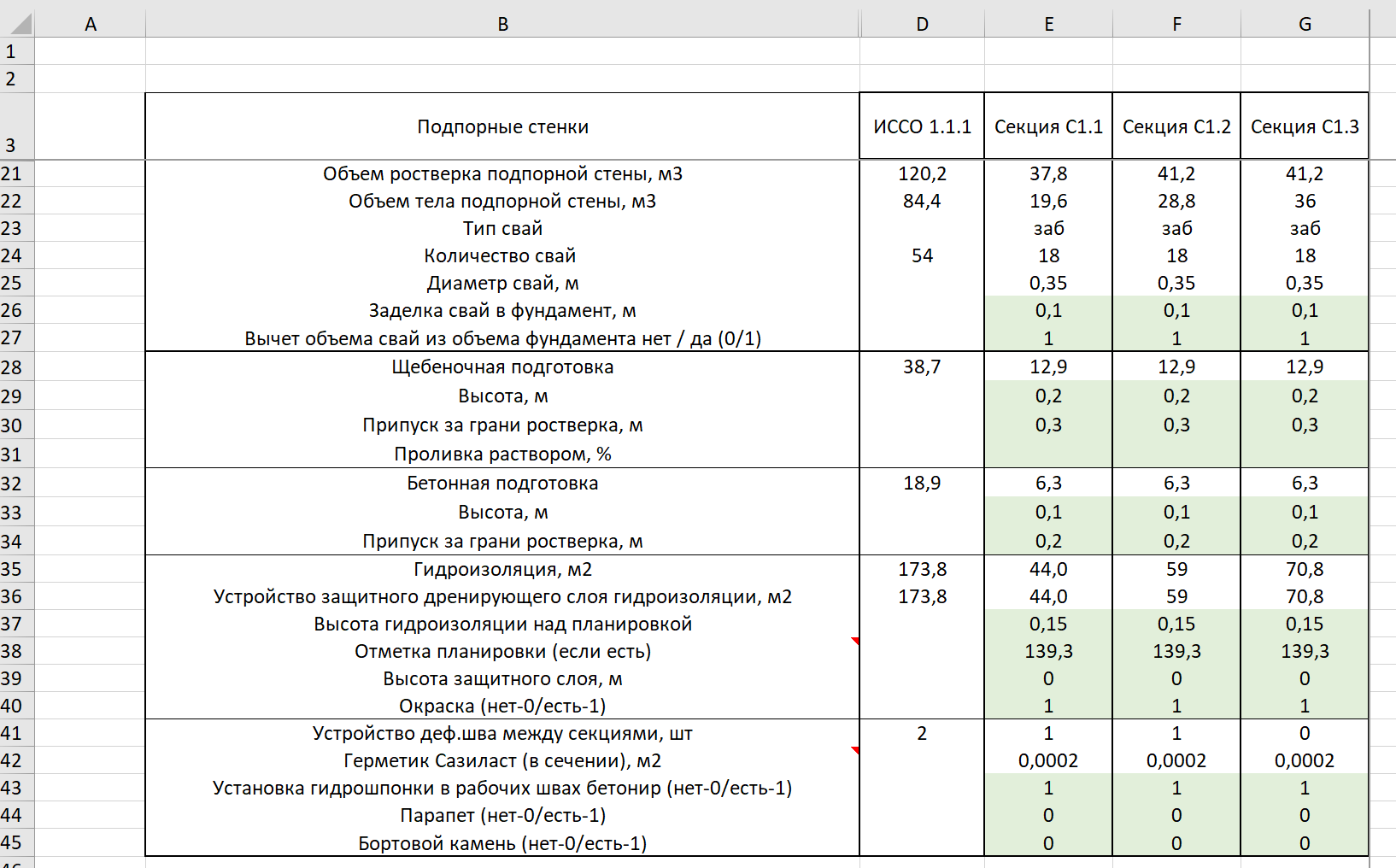


Рис.3 – Объемы работ подпорной стены и ее секций в файле Data

Объемы и параметры столбца с названием ИССО1.1.1 – содержат в себе суммирование объемов по всем секциям. Разделение объемов бетона и объемов арматуры вертикальной и фундаментных частей подпорной стены обусловлены исключительно производственной необходимостью, так как в некоторых случаях они могут иметь различную марку и характеристики бетона.

В файле Data предусмотрена связка с еще одним Excel файлом который производит расчет количества арматуры подпорной стены – ARM\_ПСТ. Все основные функции работы этого файла были уже написаны на момент разработки программы и в этой работе описываться не будут.

Мной в него вносились лишь незначительные корректировки и прописывались формулы для расчета длин стержней. Так же туда отправлялись необходимые параметры из файла Data для расчетов длин арматурных стержней по фундаментной и вертикальной части каждой секции. Все необходимые значения подгружаются автоматически при заполнении ячеек с названием конкретной секции с помощью ссылок функции ГПР.

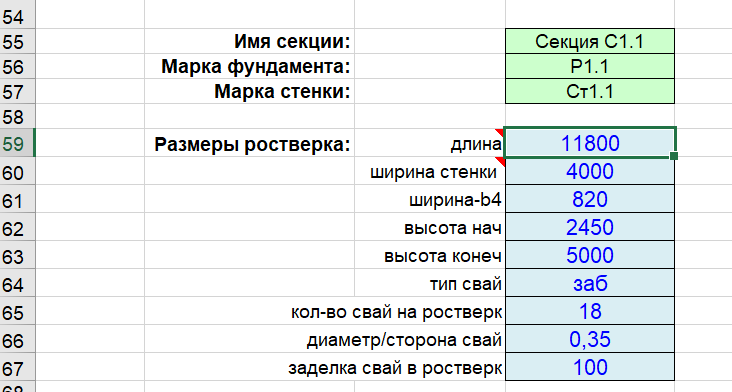


Рис. 4 – Заполнение исходных данных файла ARM\_ПСТ

Зеленым цветом отмечены ячейки, заполняемые вручную, синим – автоматически.

Заполнение происходит с помощью функции ГПР:

=ГПР($D$55;[DATA\_.xlsm]\_ПС!$E$3:$Z$25;8)\*1000

где $D$55 – ячейка, содержащее в себе название секции, по которой производится поиск столбца (секции) в файле Data, и подбор нужной строки (в данном случае 8 по счету) с необходимым параметром.

Эта связка необходима для полноценного заполнения всех необходимых объемов для Ведомости объемов работ и дальнейшего усовершенствования и развития моей программы.

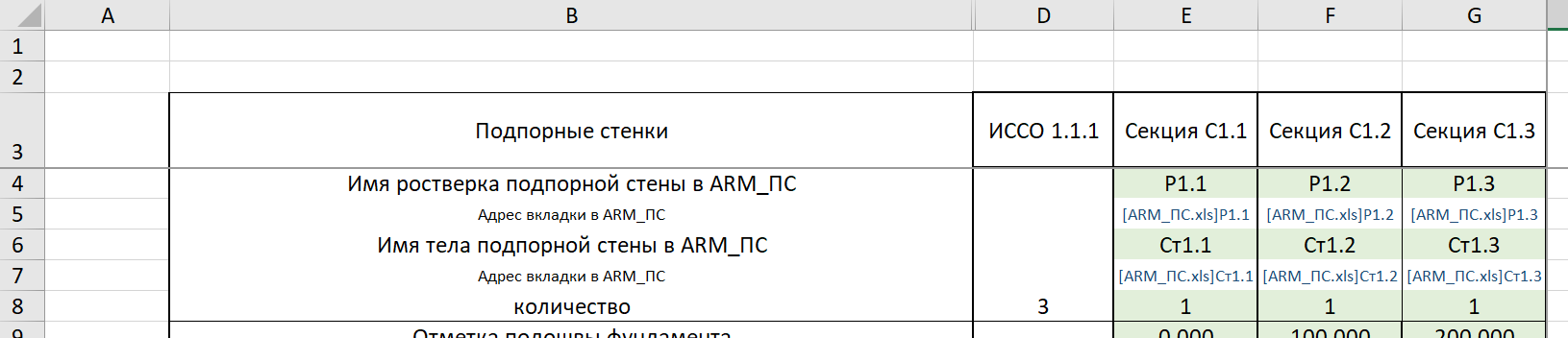


Рис.5 – Ссылки на файлы с расчетом армирования отдельных компонентов подпорной стены.

Результатом работы Excel файла ARM\_ПСТ является расчет массы всех стержней арматуры фундаментной и вертикальной частей каждой секции подпорной стены по отдельности. В файле Data они распределяются по диаметрам и классам и приводятся на странице ВОР\_ПСТ.

ГПР($D$56;'C:\Users\Ponka\Desktop\Диплом\Diplom2\[DATA\_.xlsm]\_ПС'!$E$3:$Z$25;11)\*1000

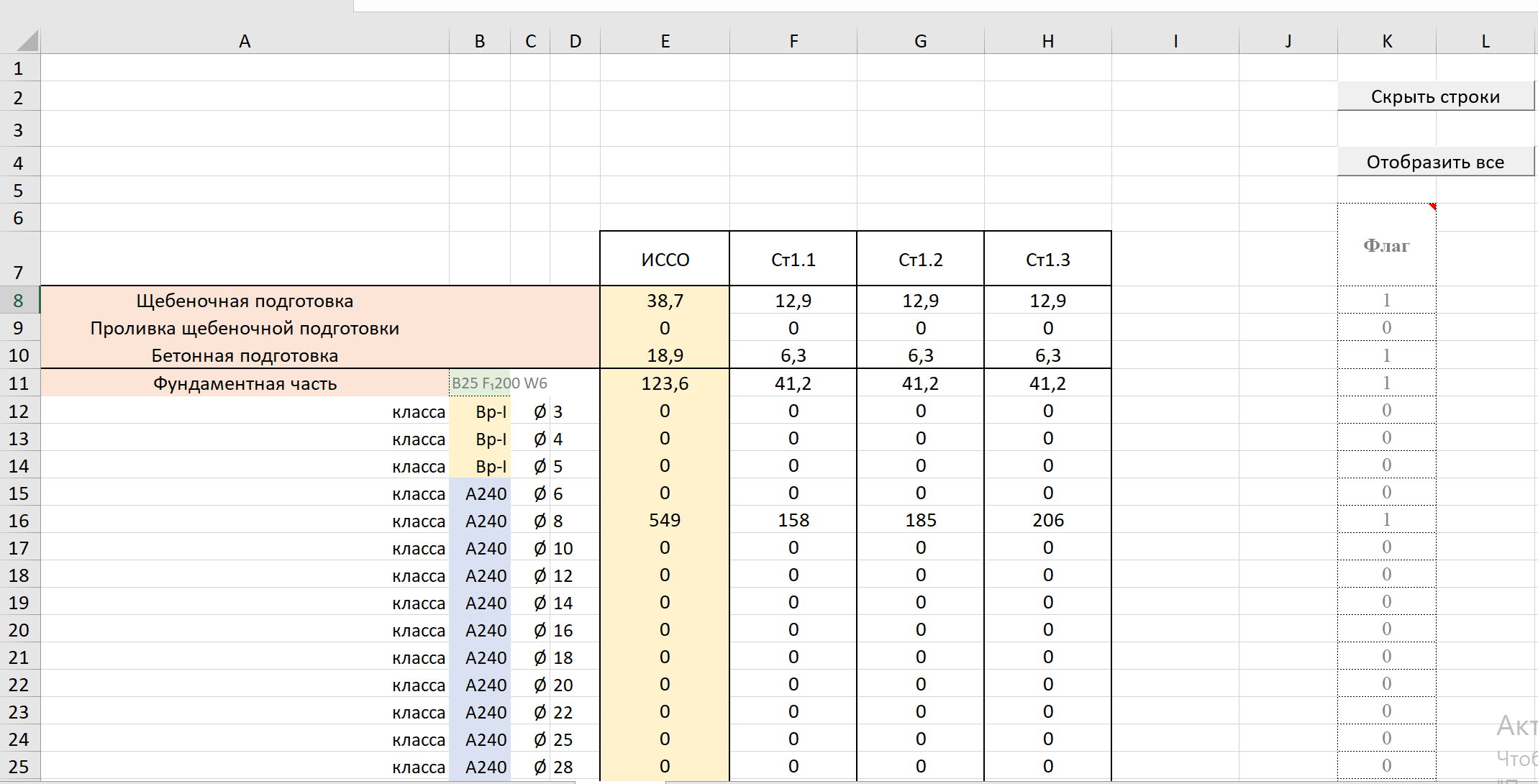


Рис. 6 – Фрагмент Таблицы объемов ВОР\_ПС из файла Data

Флаг и кнопки «скрыть строки» и «показать строки» скрывают нулевые объемы с сохранением единого шаблона заполнения, что позволяет копировать файлы E из объекта в объект и не перебивать искомые адреса ячеек. На этой логике построено все взаимодействие как между отдельными файлами Excel между собой, так и с программным кодом.

Ссылка на количество арматуры каждого диаметра и класса задается с помощью:

=ДВССЫЛ(АДРЕС(9;3;;;\_ПС!E$5))

где ПС!E$5 - ячейка в которой содержится ссылка на необходимый лист в файле ARM\_ПС, а адрес ячейки, содержащий необходимый объем всегда одинаковый благодаря структуре расчетного листа каждого элемента.

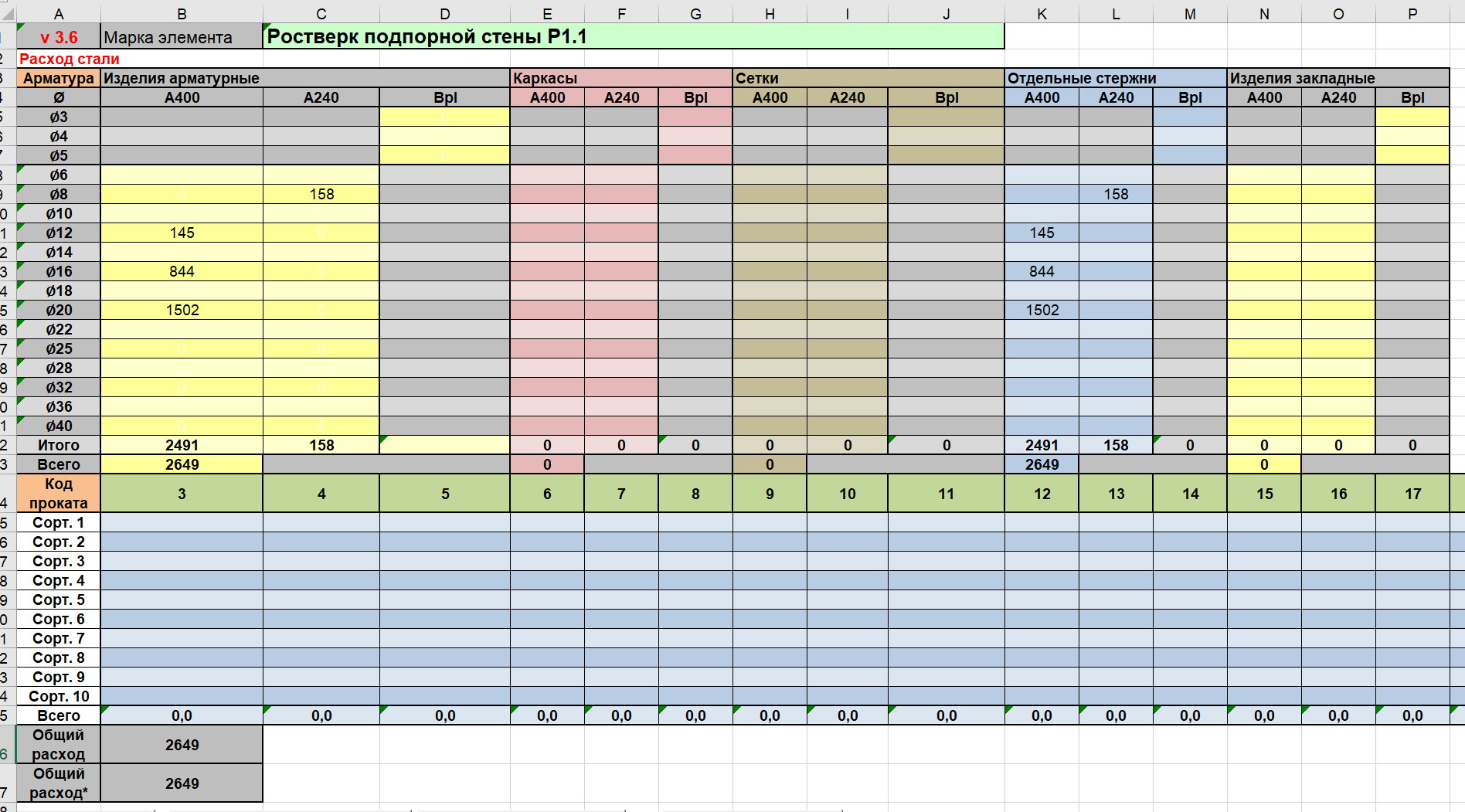


Рис.7 – Фрагмент расчетного листа\* объемов арматуры ростверка Р1.1 секции подпорной стены в файле ARM\_ПСТ

\* - не разрабатывалось в рамках магистерской диссертации.

После заполнения и подсчета всех объемов в файле Data, можно приступить к формированию одного из исходных продуктов – Ведомости объемов работ (далее ВОР). Это отдельный файл Excel , который не содержит в себе ни одной расчетной формулы, а только лишь ссылки на файл Data , в котором уже есть все необходимые исходные данные и объемы для заполнения ВОР. Это было сделано специально, для того чтобы между файлами, на основе которых производится построение и заполнение основных чертежей, и файлами Ведомостей объемов работ не допускалось никаких разночтений.

Файл ВОР не был включен в состав Data отдельным листом, а вынесен как самостоятельный документ, исходя из производственной необходимости.

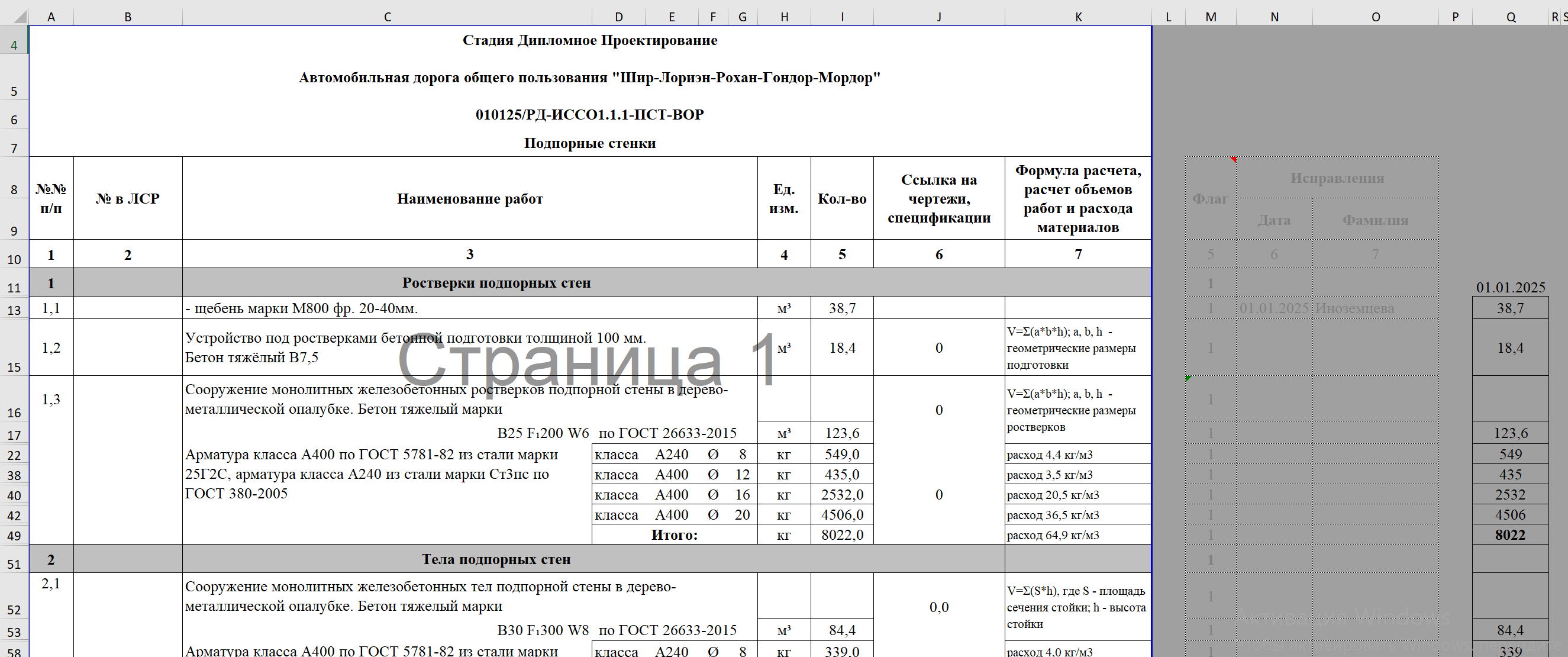


Рис.8 – Фрагмент листа из файла ВОР\_ПС

Как можно заменить, в этом файле так же предусмотрен протокол скрытия нулевых объемов. Так же можно обратить внимание на задублированные объемы работ в столбце «Q». Это было сделано для того, чтобы можно было отслеживать внесенные изменения по датам. Они не выводятся на печать и необходимы только для контроля версий. Если какой-то объем в какой-то момент времени будет отличаться от предыдущей версии, он подсветится красным цветом с помощью условного форматирования для того, чтобы обратить на это внимание пользователя и исключить вероятность случайных изменений.

**Работа с Python**

Программа делится на несколько модулей:

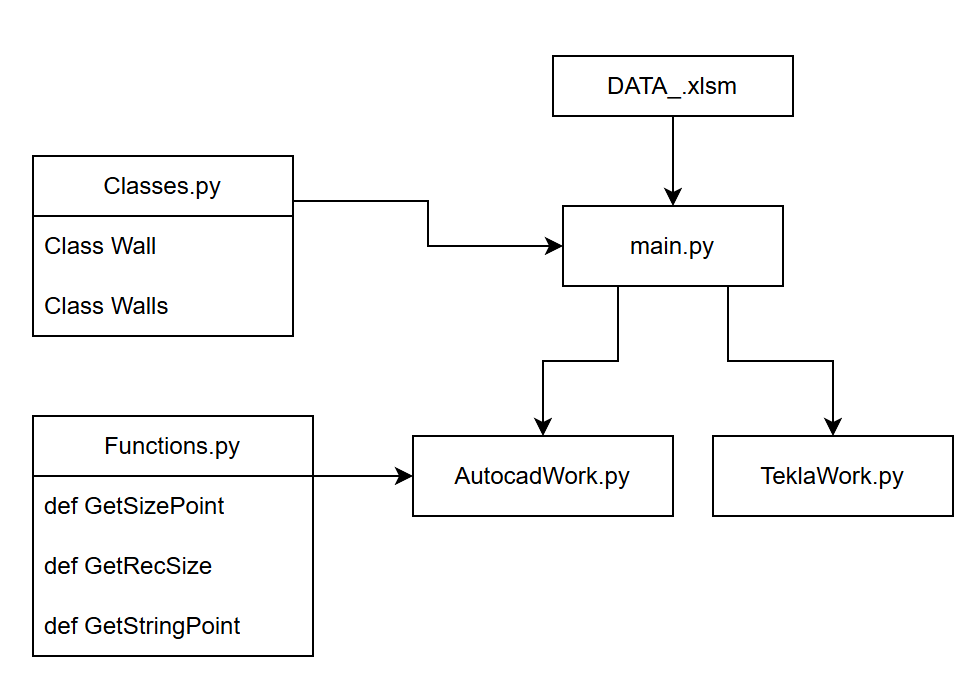


Рис.9 – Схема взаимодействия модулей программы.

Первым делом необходимо выгрузить данные для отрисовки опалубки секции подпорной стены. Для этого будет использована Python библиотека для работы с Excel – Openpyxl. Исходным является файл DATA\_.xlsm, для успешной выгрузки данных необходимо, чтобы файл находился в одной папке с исполняемым файлом и имел расширение .xlsm (из-за особенностей работы библиотеки).

Алгоритм считывания данных заключается в том, что после открытия файла исходных данных происходит последовательный проход по ячейкам начиная со стартовой и данные записываются в объект класса Wall. Число строк в столбце постоянно и каждая переменная имеет свой постоянный адрес. Число столбцов варьируется в зависимости от количества секций подпорной стены, и программа продолжает считывать их, пока не наткнется на пустой столбец.

Параллельно при вводе данных производится конвертация единиц всех линейных размеров из метров в миллиметры.

Каждая секция является объектом класса Wall, который включает в себя информацию из всех ячеек, представленных на Рис.2, описывающих положение и характеристики конкретной секции.

Программа работает со списком объектов класса Wall. Этот список хранится в классе Walls.

Листинг 1. Сlass Walls

|  |
| --- |
| class Walls(object):  name: str # Название сооружения  total\_count: int # Колличество секций в сооружении  sections = [] # Список классов(Wall) параметров секции  sections\_coors = [] # Список списков координат секций |

После выгрузки необходимой информации начинается этап отрисовки чертежа в Autocad. Для работы с Autocad была использована библиотека PyAutocad. Для начала работы модуля необходимо открыть чертеж Autocad и зайти в пространство модели, иначе программа выдаст ошибку. Первым шагом пользователю необходимо задать точку вставки - incert\_point. Программа считает координаты, введенные пользователем и запишет только координату Х, а в качестве координаты У - будет использован параметр //foundation\_base. Это связано с необходимостью выстраивать опалубочный в Балтийской системе высот и получать все дальнейшие отметки в корректном формате.

Построение видов производится исходя из топологии, она показывает какие точки должны соединиться между собой в отрезки и для всех секций одинаковая. Список координат для каждой секции тоже повторяется, а вот значения координат всегда разные.

Следующим этапом является заполнение списка координат для каждого вида опалубочного чертежа секции подпорной стенки. Для каждой секции записывается отдельный список координат с учетом заданных расстояний между видами и отсчитывается от нуля (см. Приложение ).

В качестве примера разберем фрагмент программы, который рассчитывает координаты вида 1-1

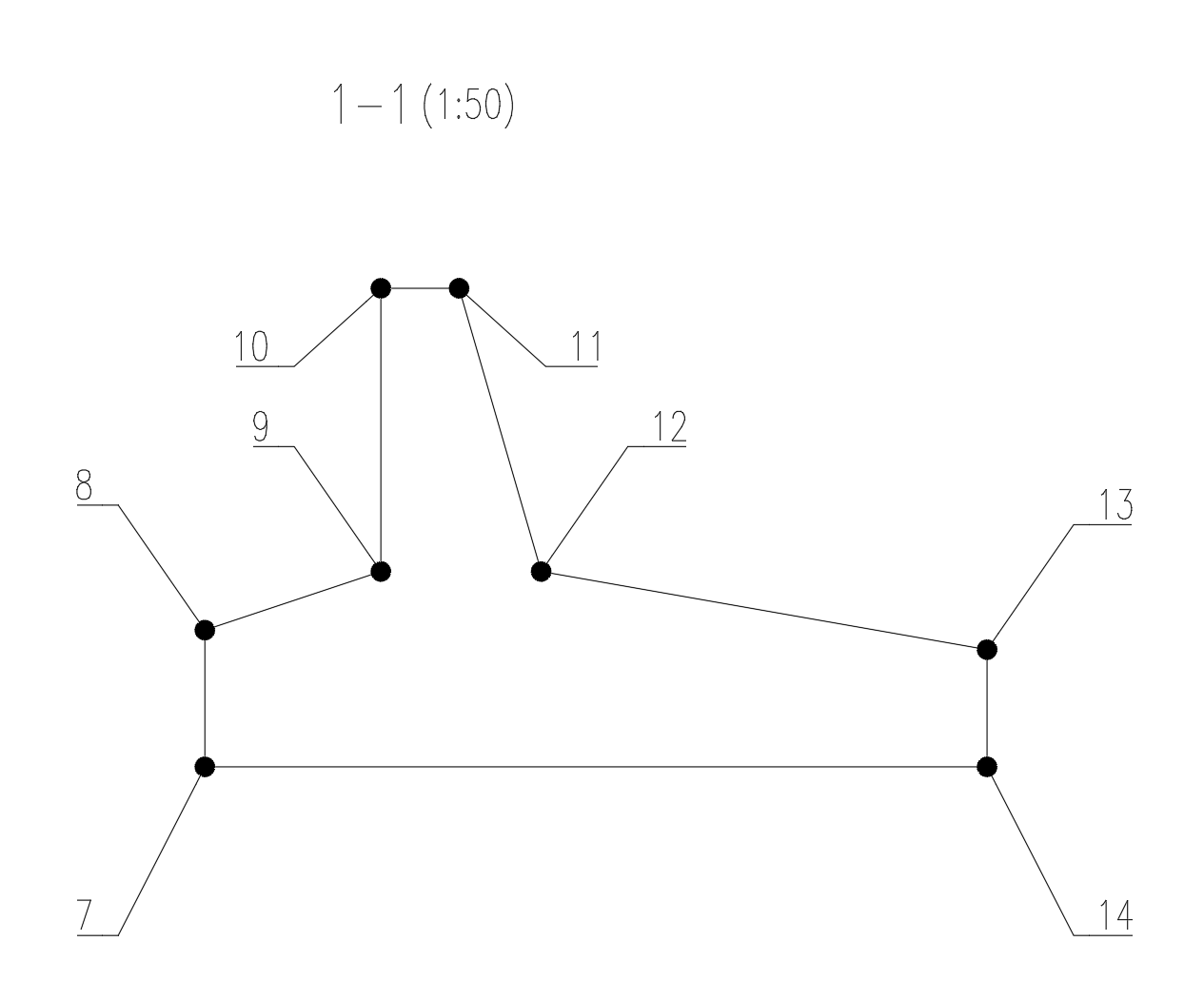


Рис.10 – Схема расстановки координат топологии вида 1-1

|  |
| --- |
| #Вид 1-1  #Задаем точку начала координат чтобы каждый вид считать от 0.0 а не прибавлять все расстояния в каждой точке  start\_coordinates = APoint(incert\_point.x + input\_data.leght + View\_l1, incert\_point.y)  SECTION\_Coor.append(APoint(start\_coordinates.x, start\_coordinates.y)) #7  SECTION\_Coor.append(APoint(start\_coordinates.x, start\_coordinates.y + input\_data.t2)) #8  SECTION\_Coor.append(APoint(start\_coordinates.x + input\_data.edge\_distance , start\_coordinates.y + input\_data.t1)) #9  SECTION\_Coor.append(APoint(start\_coordinates.x + input\_data.edge\_distance , start\_coordinates.y + input\_data.height\_start)) #10  SECTION\_Coor.append(APoint(start\_coordinates.x + input\_data.edge\_distance + input\_data.top\_wall\_width, start\_coordinates.y + input\_data.height\_start)) #11  SECTION\_Coor.append(APoint(start\_coordinates.x + input\_data.edge\_distance + input\_data.bottom\_wall\_width, start\_coordinates.y + input\_data.t3)) #12  SECTION\_Coor.append(APoint(start\_coordinates.x + input\_data.foundation\_width, start\_coordinates.y + input\_data.t4)) #13  SECTION\_Coor.append(APoint(start\_coordinates.x + input\_data.foundation\_width, start\_coordinates.y)) #14 |

Листинг 2. Расчет координат вида 1-1

Для каждого вида задается своя точка start\_coordinates. Это необходимо для того чтобы следующие координаты вида можно было отсчитывать не от общего нуля incert\_point, а от своего локального – левой нижней точки вида. Каждая точка start\_coordinates учитывает расстояние между видами внутри модели. Это расстояние задается с помощью переменных внутри кода программы.

Листинг 3. Переменные, описывающие расстояния между видами.

|  |
| --- |
| Line\_distance = 85000 #расстояние между двух видовых рамок секций  View\_l1 = 5000 #расстояние между фасадом и сечением 1-1 (по горизонтали)  View\_l2 = 5000 #расстояние сечением 1-1 и сечением 2-2 (по горизонтали)  View\_l3 = 10000 #расстояние между фасадом и планом (по вертикали) |

Далее производится заполнение массива топологии списками, содержащими в себе начало и конец отрезка.

Листинг 4. Заполнение массива топологии вида 1-1

|  |
| --- |
| Topology = []  #Вид 1-1  Topology.append([7,8])  Topology.append([8,9])  Topology.append([9,10])  Topology.append([10,11])  Topology.append([11,12])  Topology.append([12,13])  Topology.append([13,14])  Topology.append([14,7]) |

После составления списка координат и топологии начинается отрисовка видов с простановкой необходимых размеров и текста.

Каждый элемент чертежа должен располагаться на своем слое. Это необходимо для соблюдения стиля печати каждого элемента (контурные линии выводятся толще, чем размерные и т.д.). Чтобы слой мог быть выбран – он должен уже существовать в модели.

Для переключения слоя используется переменная:

acad.doc.ActiveLayer = acad.doc.Layers.Item ("Contur")

Отрисовка делится на несколько этапов:

1. Отрисовка контура видов. Производится в слое “Contur”

Отрисовка производится отрезками с помощью метода:

acad.model.AddLine (start\_point, end\_point)

метод принимает начальную и конечную точку отрезка, которые являются объектами класса APoint – класс библиотеки PyAutocad.

1. Простановка размеров и отметок уровня. Производится в слое “Size”

Отрисовка размерный линий производится в двух масштабах (1:100; 1:50), соответственно в двух размерных стилях 'LIN100' и 'LIN50'. Для корректного вывода я создала функцию GetSizePoint. Она принимает две координаты – начало и конец размерной линии, направление размера (задается через угол в радианах) и местоположение размерной линии (задается длиной). Результатом работы функции является вывод трех координат, необходимых для простановки размера.

Отрисовка размерных линий производится с помощью метода AddDimRotated, который принимает три координаты и угол направления размера. Это необходимо для более точного позиционирования размера в режиме линейной отрисовки. Метод, принимающий только три точки – отрисовывает параллельный размер.

Простановка отметок уровня производится вставкой в необходимые точки, заранее созданного в автокаде аннотативного блока отметки 'Otmetka'. Поскольку блок аннотативный – менять его масштаб в зависимости от вида нельзя. Он вставляется в чертеж только в масштабе 1:1. Далее уже производится настройка списка его аннотативных масштабов уже внутри автокада, если требуется. Поэтому при вставке, если список аннотативных масштабов не настроен, блок отметки может оказаться очень маленьким и незаметным на чертеже. Вставка блока производится с помощью метода: acad.model.InsertBlock().

1. Создание таблицы ведомости объемов работ. Производится в слое “T\_Border”

Первым этапом создается сама таблица. Стоит отметить что первая строка таблицы создается сразу объединенной с помощью метода: acad.model.AddTable().

Позиционируется таблица в правом верхнем углу ограничивающей рамки в пространстве модели. Далее пользуемся методами: tabtable.SetColumnWidth (0, 14500), table.SetRowHeight (0, 1500) для редактирования размеров строк и столбцов, поскольку при создании таблицы – задать эти параметры нельзя. По тому же принципу производится заполнение таблицы и выставление стиля и высоты текста. Для заполнения таблицы применяется метод:

tabtable.SetText (1, 1 , 'Ед.изм.').

Далее циклом проходимся по всей таблице для того, чтобы отредактировать высоту и стиль текста.

Листинг 5. Редактирование высоты текста и стиля в таблице

|  |
| --- |
| table\_text\_height=350  for row in range(table.Rows):  for col in range(table.Columns):  table.SetCellTextStyle(row, col, "RS0.7")  table.SetCellAlignment(row, col,5)  table.SetCellTextHeight(row, col, table\_text\_height) |

1. Простановка названий видов. Производится в слое “Text\_B”

Названия видов выводятся просто многострочным текстом на заданном расстоянии от вида с помощью метода: acad.model.AddMText(). Параметры текста так же задаются отдельно.

Листинг 6. Редактирование высоты текста и стиля в многострочном тексте

|  |
| --- |
| acad.doc.ActiveLayer = acad.doc.Layers.Item("Text\_B")  text\_style = acad.doc.TextStyles.Item("RS0.7")  acad.doc.ActiveTextStyle = text\_style  to\_view\_dis = 2500 # Расстояние от вида до его текста |

1. Отрисовка рамки – границы всех чертежей конкретной секции. Производится в слое “Notes”, не выводящимся на печать.

Рамка рисуется отрезками, но для того чтобы их отрисовать, сначала необходимо рассчитать все необходимые размеры и координаты. Для этого я создала функцию “ GetRecSize ”, которая принимает размеры отступов границ рамки от видов и максимальные размеры самих видов, после чего высчитывает размеры для отрезков.

Листинг 6. Функция расчета параметров рамки

|  |
| --- |
| def GetRecSize(incert\_point, max\_width, max\_height):  vertical\_otstyp = 15000  horizontal\_otstyp\_left = 10000  horizontal\_otstyp\_right = horizontal\_otstyp\_left \* 3  H = max\_height + vertical\_otstyp \* 2  B = max\_width + horizontal\_otstyp\_left + horizontal\_otstyp\_right  Point = APoint(incert\_point.x - horizontal\_otstyp\_left, incert\_point.y - max\_height - vertical\_otstyp )  rez = [Point, H, B]  return rez |

Дополнительно стоит отметить, что в программа учитывает переменную высоту подпорной стенки при отрисовки видов. Если у секции постоянная высота, то программа не отрисует вид 2-2 потому что в нем нет необходимости.

На этом заканчивается работа с Autocad. Далее программа завершает свою работу. Для того чтобы начать работу с Tekla Structures, необходимо инициализировать повторный вызов программы.

Первым этапом работы модуля TeklaWork.py является подключение библиотек. Для корректной работы программы файлы библиотеки API Tekla: Tekla.Structures.dll, Tekla.Structures.Drawing.dll, Tekla.Structures.Model.dll- должны находиться в папке вместе с остальными файлами программы. Для подключения библиотек, первым делом необходимо получить полный путь к файлу проекта. Для этого из библиотеки pathlib импортируется класс Path из которого используется метод Path.cwd().

После чего вручную прописывается “адресс” каждого dll-файла: clr.AddReference(str(tekla\_path)+ "/Tekla.Structures.dll"). Далее производится импорт необходимых элементов модели из библиотек таких как: модель, балка, точка, положение и т.д.

Функция, отвечающая за отрисовку в Tekla, называется InputIKomponent(). Она принимает объект класса Walls - список секций подпорной стенки. Для начала отрисовки компонентов необходимо сначала подключиться к пространству модели. В программе предусмотрена проверка на наличие Tekla на компьютере пользователя и успешное подключение к модели.

Листинг 7. Подключение к пространству модели в Tekla Structures

|  |
| --- |
| def InputIKomponent(input\_datas):  model = Model()  if not model.GetConnectionStatus():  print("Не удалось подключиться к модели Tekla")  exit() |

Следующим этапом необходимо циклом пройтись по каждой секции и создать в Tekla секцию подпорной стены. Для этого необходимо создать элемент Beam сзаданным стандартным профилем RCRW. Профиль можно изменить, отредактировав каждый параметр по отдельности в строке описания профиля, где каждая цифра обозначает определенный размер поперечного сечения:

RCRW5000\*4000-400\*900-820-1145\*700-1000\*600

При вставке балки, в программе предусмотрена проверка на успешную вставку. Результат проверки выводится пользователю на экран.

Для позиционирования балки в пространстве модели, используются две координаты – начальная и конечная по центру балки. Они заданы в Excel – файле Data. Положение балки по глубине задается отдельным параметром.

Листинг 8. Установка положения балки по глубине в позицию "Спереди"

|  |
| --- |
| position = getattr(Position.DepthEnum, 'FRONT')  beam.Position.Depth = position |

При вставке, балка создается с постоянной высотой, что не всегда соответствует действительности. Поэтому необходимо задать обрезку балки по высоте. Для этого необходимо вставить еще одну балку большего сечения, которая будет выступать обрезающим объектом, а в последствии антителом.

**Перспективное развитие программы и ее дальнейшее применение**

Эта программа разработана для автоматизации процесса выпуска рабочей документации по комплектам монолитных железобетонных подпорных стен уголкового типа на базе проектного института Гипростроймост.

В первую очередь она предназначена для сокращения времени на работу с рутинными задачами, которые циклично повторяются в больших количествах и практически не претерпевают изменений. В частности, это отрисовка опалубочных чертежей. Опалубочные чертежи — это детализированные схемы, по которым изготавливается опалубка для бетонирования подпорной стены. Они должны точно отражать геометрию конструкции и ее положение в пространстве, расположение закладных элементов и технологические особенности бетонирования.

Подпорные стены часто имеют типизированное поперечное сечение лишь с небольшими изменениями (высота, длина, отметка подошвы фундамента, армирование). Ручное черчение занимает много времени и повышает риск ошибок, связанных с не одновременным внесением изменений в различные составные части проекта. Можно легко поменять геометрию секции в чертеже Autocad, но не изменить эти данные в ведомости объемов работ и наоборот. В таком случае, данные на разных листах проекта будут отличаться друг от друга и придется затрачивать дополнительные усилия на перепроверку и установление достоверных значений.

Таким образом - основным посылом создания этой программы является желание не только увеличить скорость разработки документации, но стремление добиться соответствия между собой всей информации, представленной в проекте. Так же разработка всех последующих схожий проектов на основе одной программы позволит добиться полного единообразия оформления чертежей по всему проектному институту и не допускать разночтений в пределах одного объекта строительства.

В дальнейшем я вижу развитие разработанной программы в добавлении отрисовки не только опалубочных, но и арматурных чертежей. Вся информация, необходимая для определения размера и позиционирования каждого стержня внутри контуров подпорной стены, уже содержится в данном проекте. Конфигурацию они имеют практически идентичную от стенки к стенке и могут нуждаться лишь в незначительной корректировке вручную. На данном этапе разработки данная функция не рассматривалась ввиду большей детализации арматурного чертежа по сравнению с опалубочным, что влечет за собой большую трудоемкость разработки.

Еще одним этапом доработки я вижу возможность добавления в проект возможности позиционирования и отрисовки свайного поля. Да, подпорные стенки тоже могут быть со свайным фундаментом. В чем заключается практический интерес автоматизации? В возможности автоматической расстановки свай с соблюдением необходимых требований СП 22.13330.2016 – «Основания зданий и сооружений» и учетом необходимых расчетных характеристик.

Так же я бы хотела в будущем развить эту программу и добавить в нее отрисовку ростверков и опор, так как они тоже имеют практически идентичную конфигурацию и в одном отдельном сооружении их может быть довольно большое количество.

Так же хотелось бы добавить в программу возможность работы с динамическими блоками и напрямую обращаться и записывать их атрибуты. Это позволит заметно расширить функционал и добавить отрисовку таких вещей как: обозначение вида сечения, мультивыноски и прочие динамические блоки.

**Приложения:**

Приложение 1.

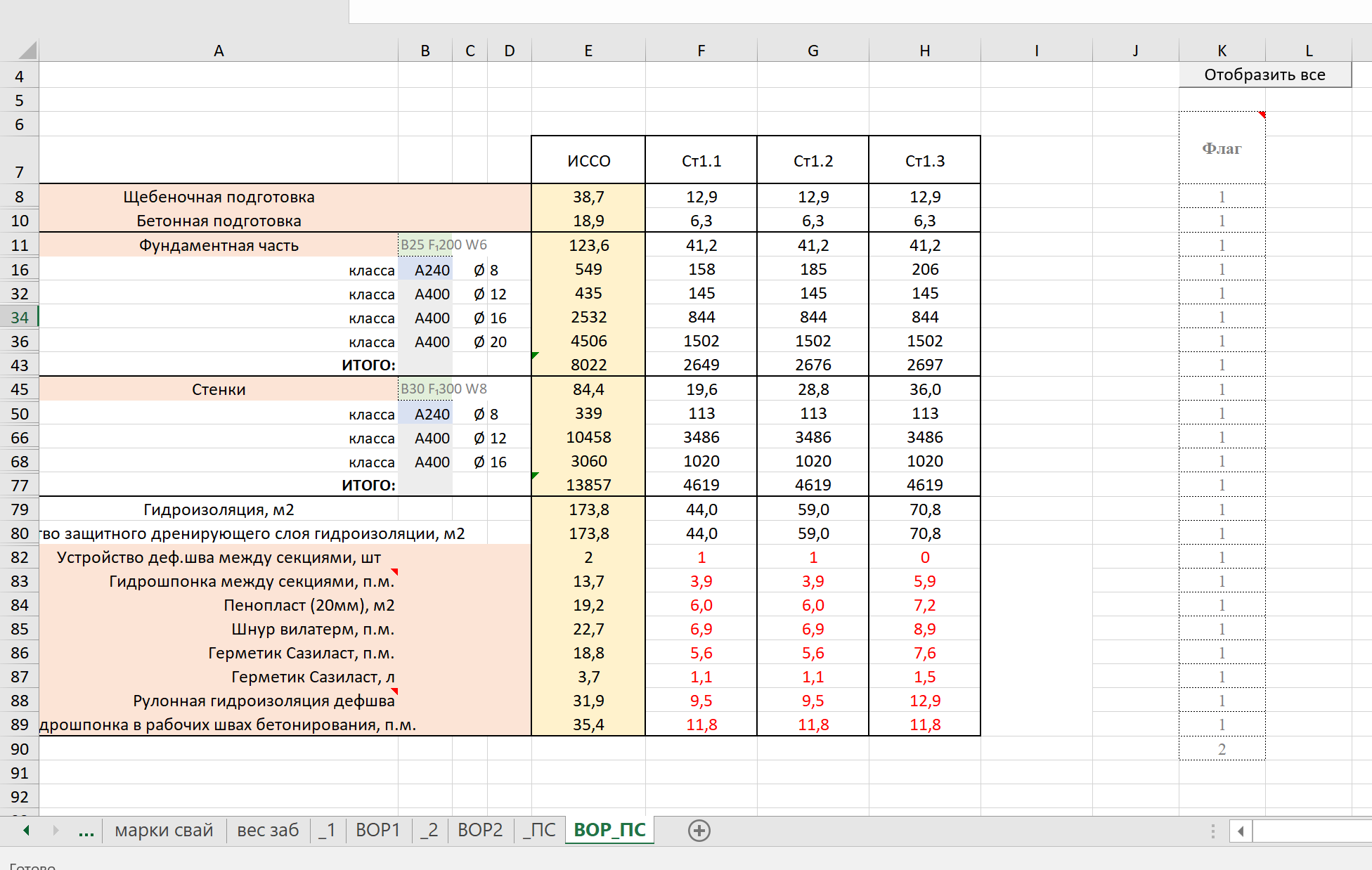
Общий вид вкладки \_ПС в Excel файле Data со всеми исходными данными\* и параметрами искусственного сооружения ИССО1.1.1 и его составляющих.



\* - зеленые ячейки заполняются вручную, белые – вычисляются автоматически.

Приложение 2.

Общий вид вкладки ВОР\_ПС в Excel файле Data со всеми расчетными и ссылочными значениями объемов.



Приложение 3.

Общий вид вкладки Р1.1 в Excel файле\* ARM\_ПСТ - расчетного листа объемов арматуры ростверка Р1.1 секции подпорной стены.



\* - не разрабатывалось в рамках магистерской диссертации, но послужило вдохновением для ее создания.

Приложение 4.

Общий вид вкладки ВОР в Excel файле ВОР\_ПСТ.

Является готовым документом, подлежащим обязательному приложению в состав томов рабочей документации.



Приложение 5.

Топология всех видов опалубочных чертежей.

Приложение 6.

Общий вид макета чертежа в пространстве модели с нанесением всех вспомогательных размеров и отметок.

