**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**КАФЕДРА САПР**

**Практическая работа №2**

**«Консольная программа для работы с CSV и XML файлами»**

**по дисциплине**

**«Объектно-ориентированное программирование»**

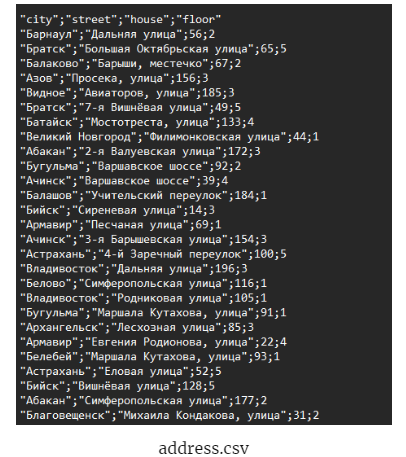
|  |  |  |
| --- | --- | --- |
| Студентки гр. 3354 |  | Чикарёва М.Д.  Запольская Н.Д. |
| Преподаватель |  | Кулагин М.В. |

**Санкт-Петербург**

**2024**

1. Задание.

Даны 2 файла-справочника городов: один в формате xml, второй - csv. Необходимо разработать консольное приложение для работы с ними. Информация в файлах представлена так:



После запуска приложение ожидает ввода пути до файла-справочника, либо команды на выход из программы (exit). По команде завершения работы приложение завершает работу.

После ввода пути до файла-справочника приложение выводит следующую информацию:

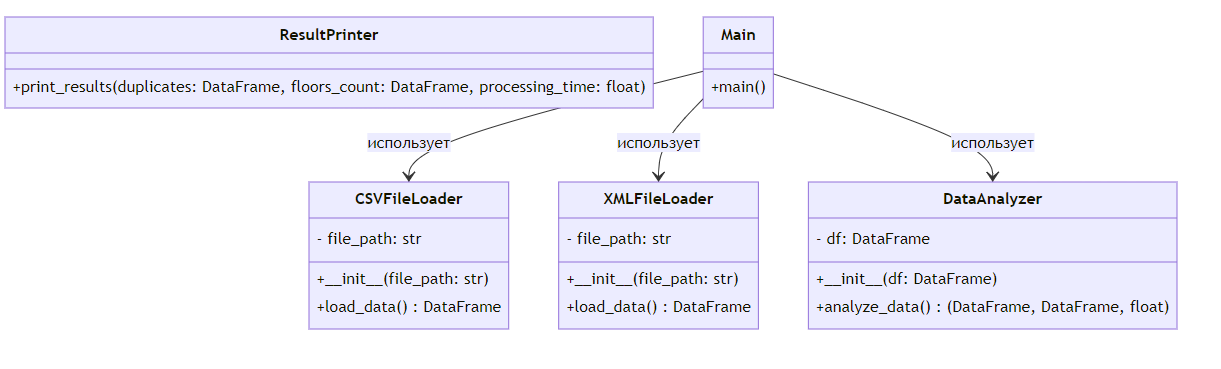
- в первой строке выводятся дублирующиеся записи с количеством повторений;

- во второй строке выводится, сколько в каждом городе: 1, 2, 3, 4 и 5-этажных зданий;  
 - показывает время обработки файла в конце.

После вывод всей информации приложение снова ожидает ввода пути до файла-справочника, либо команды на завершение работы.

В процессе работы приложение падать не должно, выход только по команде на заверешение работы.

1. Спецификация программы (алгоритм работы).



Основные компоненты программы:

1. Импорт библиотек.

Импортируются модули csv, xml.etree.ElementTree, pandas, os, time - нужны для работы с файлами, обработки данных и подсчёта времени. google.colab.drive - нужно для монтирования гугл-диска в среде гугл-коллаб.

1. Классы:

- CSVFileLoader:

Метод \_\_init\_\_(file\_path) — инициализация с путем к CSV файлу.

Метод load\_data() — загрузка данных из CSV файла в DataFrame.

- XMLFileLoader:

Метод \_\_init\_\_(file\_path) — инициализация с путем к XML файлу.

Метод load\_data() — парсит XML и загружает данные о городах, улицах, домах и этажах из XML файла в DataFrame.

- DataAnalyzer:

Метод \_\_init\_\_(df) — инициализация с DataFrame.

Метод analyze\_data() — анализ данных, поиск дубликатов и подсчет этажности.

- ResultPrinter:

Статический метод print\_results(duplicates, floors\_count, processing\_time) — вывод результатов анализа.

1. Функция main:

- Считывание пути к файлу.

- Проверка наличия файла.

- Выбор класса загрузчика в зависимости от формата файла (CSV или XML).

- Загрузка данных в DataFrame.

- Анализ данных.

- Печать результатов.

- Обработка возможных ошибок (неправильный формат файла и другие исключения).

Таким образом, программа последовательно выполняет загрузку данных из файлов, их анализ и вывод результатов на экран.

1. Описание интерфейса пользователя программы.

- При запуске программы пользователя встречает надпись с просьбой ввести путь к файлу-источнику.



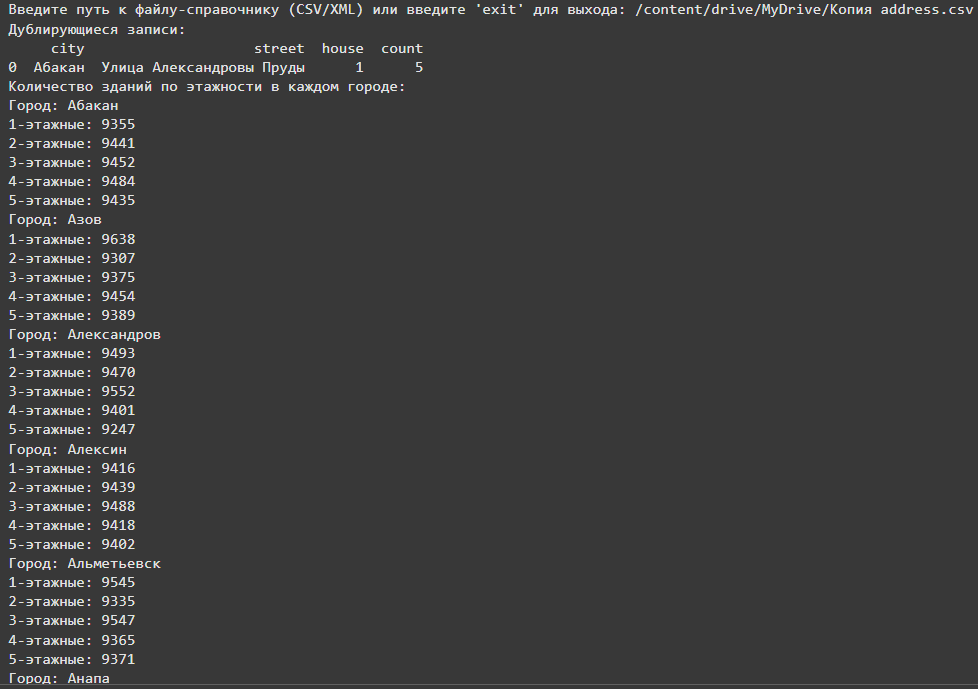
- Если вводится файл любого другого формата, кроме xml и csv, то выводится сообщение о том, что данный формат не поддерживается и нужно ввести либо xml, либо csv.



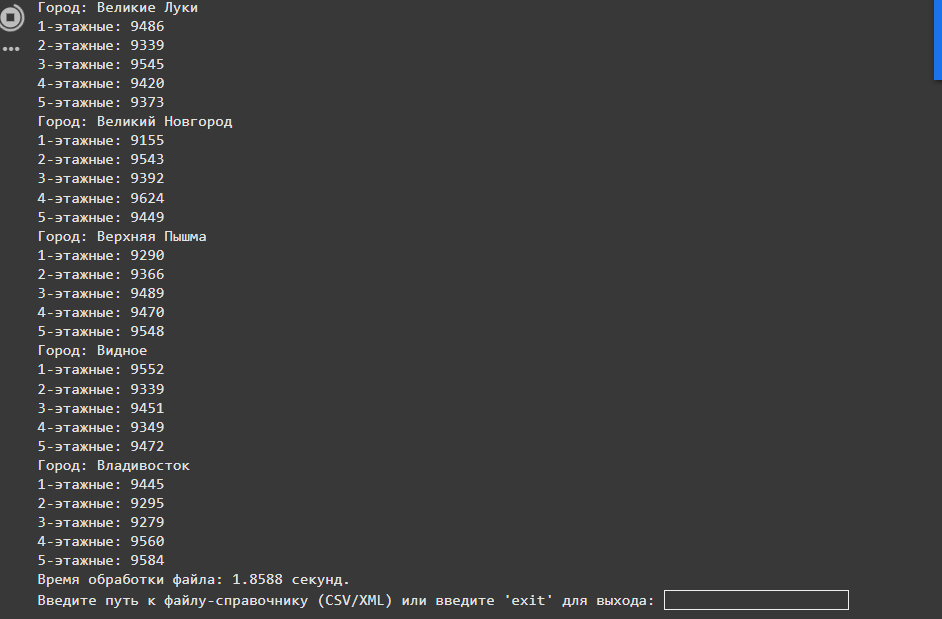
- Если неправильная ссылка или файла нет, то будет выведена информация о том, что файл не найден и нужно попробовать снова.



- Если же всё введено верно, то выведется вся нужная информация: дублирующиеся строки, сколько в каждом городе 1, 2, 3, 4 и 5-этажных зданий, а также время обработки файла.



..........................................длинный список городов..................................



1. Код программы.

import csv

import xml.etree.ElementTree as ET

import pandas as pd

import os

import time

from google.colab import drive

drive.mount('/content/drive')

class CSVFileLoader:

    def \_\_init\_\_(self, file\_path):

        self.file\_path = file\_path

    def load\_data(self):

        """Загружает данные из CSV файла в DataFrame."""

        return pd.read\_csv(self.file\_path, sep=';')

class XMLFileLoader:

    def \_\_init\_\_(self, file\_path):

        self.file\_path = file\_path

    def load\_data(self):

        """Загружает данные из XML файла в DataFrame."""

        data = []

        for event, elem in ET.iterparse(self.file\_path, events=('end',)):

            if elem.tag == 'item':

                city = elem.get('city')

                street = elem.get('street')

                house = elem.get('house')

                floor = int(elem.get('floor'))

                data.append({city, street, house, floor})

                elem.clear()

        return pd.DataFrame(data)

class DataAnalyzer:

    def \_\_init\_\_(self, df):

        self.df = df

    def analyze\_data(self):

        start\_time = time.time()

        duplicates = self.df.groupby(['city', 'street', 'house']).size()

        duplicates = duplicates[duplicates > 1].reset\_index(name='count')

        floors\_count = self.df.groupby(['city', 'floor']).size().unstack(fill\_value=0)

        processing\_time = time.time() - start\_time

        return duplicates, floors\_count, processing\_time

class ResultPrinter:

    @staticmethod

    def print\_results(duplicates, floors\_count, processing\_time):

        if not duplicates.empty:

            print("Дублирующиеся записи:")

            print(duplicates)

        else:

            print("Дублирующихся записей не найдено.")

        print("Количество зданий по этажности в каждом городе:")

        for city, counts in floors\_count.iterrows():

            print("Город: {}".format(city))

            for floor, count in counts.items():

                print("{}-этажные: {}".format(floor, count))

        print("Время обработки файла: {:.4f} секунд.".format(processing\_time))

def main():

    print("Введите путь к файлу-справочнику (CSV/XML) или введите 'exit' для выхода.")

    while True:

        file\_path = input("Введите путь к файлу-справочнику (CSV/XML) или введите 'exit' для выхода: ").strip()

        if file\_path.lower() == 'exit':

            print("Завершение работы.")

            break

        if not os.path.isfile(file\_path):

            print("Файл не найден. Попробуйте снова.")

            continue

        try:

            if file\_path.lower().endswith('.csv'):

                loader = CSVFileLoader(file\_path)

            elif file\_path.lower().endswith('.xml'):

                loader = XMLFileLoader(file\_path)

            else:

                raise ValueError("Неподдерживаемый формат файла. Пожалуйста, используйте CSV или XML.")

            df = loader.load\_data()

            analyzer = DataAnalyzer(df)

            duplicates, floors\_count, processing\_time = analyzer.analyze\_data()

            ResultPrinter.print\_results(duplicates, floors\_count, processing\_time)

        except ValueError as ve:

            print("Ошибка: {}".format(ve))

        except Exception as e:

            print("Произошла ошибка: {}".format(e))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

1. Выводы.

Программа работает верно при всех возможных ситуациях. Если неверный формат или файл не найден, то программа не вылетает, а выводит сообщение, что не так, и даёт сразу же поле для повторного ввода. Если же файлы верные, то вся информация будет выведена на экран. После завершения программа не завершается, она выйдет только в случае ввода ‘exit’.

Разработанны программный код был собран в Google Colaboratory, а также выложен на GitHub.

<https://github.com/essslish/oop_chi_and_zap.git>