МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

**Отчет по курсовой работе**

по дисциплине «Программная инженерия»

по теме:

сбор и подготовка данных о курсе валют с сайта Центрального Банка России

Работу выполнил

студент группы А-13б-20

Бегунов Никита

Научный руководитель:

Кожевников Антон Вадимович

Преподаватель:

Маран Михкель Михкелевич

Москва 2022

Оглавление

[1 Введение 3](#_Toc122704946)

[2 Описание предметной области веб-скрейпинга 3](#_Toc122704947)

[2.1 Принцип работы и область применения веб-скрейпинга 3](#_Toc122704948)

[2.2 Постановка задачи парсинга сайта Центрального Банка России 4](#_Toc122704949)

[2.3 Сбор информации с сайта Центрального Банка России 5](#_Toc122704950)

[3 Описание работы приложения 6](#_Toc122704951)

[3.1 Описание разработанного функционала 6](#_Toc122704952)

[3.2 Используемый язык и библиотеки 6](#_Toc122704953)

[3.3 Архитектура приложения 7](#_Toc122704954)

[3.4 Алгоритм работы программы 8](#_Toc122704955)

[3.5 Параметры запуска приложения 8](#_Toc122704956)

[4 Проверка работоспособности приложения 8](#_Toc122704957)

[5 Заключение 10](#_Toc122704958)

[6 Список использованных источников 10](#_Toc122704959)

[7 Приложения 12](#_Toc122704960)

[7.1 Приложение 1. Файл Parsers.py 12](#_Toc122704961)

[7.2 Приложение 2. Файл RowFromTable.py 13](#_Toc122704962)

[7.3 Приложение 3. Файл Savers.py 13](#_Toc122704963)

[7.4 Приложение 4. Файл main.py 16](#_Toc122704964)

# 1 Введение

Сейчас интернет – самый большой источник информации в мире. И этой информации становится все больше с каждым днем, поэтому важно уметь быстро необходимую информацию из огромного массива всего того, что люди видят собственными глазами. И тем более полезно, когда эту работу выполняет компьютер. Для выполнения такой работы была изобретена технология веб-скрейпинга, когда программа под видом обычного пользователя заходит на сайт и собирает необходимую информацию. Частным случаем веб-скрейпинга является парсинг. В этом случае не требуется эмуляция действий пользователя - программа просто открывает необходимую страницу и берет необходимую информацию.

Целью данной работы является разработка приложения, способного собирать данные о курсах валют за различные периоды времени. В рамках данной работы необходимо разработать и реализовать приложение, выполняющее следующий ряд функций:

* Осуществление парсинга сайта Центрального Банка России (cbr.ru) и получение информации о валюте, заданной пользователем, за заданный период времени;
* Подготовка информации для дальнейшего обучения на ее основе нейронной сети по предсказанию курса валют;
* Сохранение полученных данных в файле с расширением, необходимым пользователю.

Собирать данную информацию вручную трудозатратно, долго и дорого, что заставляет прибегнуть к автоматизированным программным средствам. Более того, с их помощью можно автоматически ежедневно получать актуальную информацию для любых мировых валют.

# 2 Описание предметной области веб-скрейпинга

## 2.1 Принцип работы и область применения веб-скрейпинга

Парсинг (Parsing) – это принятое в информатике определение синтаксического анализа. Для этого создается математическая модель сравнения лексем с формальной грамматикой, описанная одним из языков программирования. Программа (скрипт), дающая возможность компьютеру «читать» - сравнивать предложенные слова с имеющимися во Всемирной сети, называется парсером. Сфера применения таких программ очень широка, но все они работают практически по одному алгоритму.

Сбор информации в интернете – трудоемкая, рутинная, отнимающая много времени работа. Парсеры, способные в течение суток перебрать большую часть веб-ресурсов в поисках нужной информации, автоматизируют ее. Наиболее активно «парсят» всемирную сеть роботы поисковых систем. Но информация собирается парсерами и в частных интересах. На ее основе, например, можно написать диссертацию. Парсинг используют программа автоматической проверки уникальности текстовой информации, быстро сравнивая содержимое сотен веб-страниц с предложенным текстом.

Независимо от того, на каком формальном языке программирования написан парсер, алгоритм его действия остается одинаковым:

1. Выход в интернет, получение доступа к коду веб-ресурса и его скачивание.
2. Чтение, извлечение и обработка данных.
3. Представление извлеченных данных в удобоваримом виде.

Для анализа заданного текста такое программное обеспечение обычно использует отдельный лексический анализатор. Она называется токенайзером или лексером. Токенайзер разбивает все входные данные на токены – отдельные символы, например, слова. Полученные таким образом токены служат входными символами для парсера. Затем программа обрабатывает грамматику входных данных, анализирует и создает синтаксическое дерево. На этой основе идет дальнейшая работа парсера с информацией – генерация кода или выборка по определенным критериям.

Существуют два основных метода парсинга: нисходящий и восходящий. Обычно они различаются порядком, в котором создаются узлы синтаксического дерева.

* Сверху-вниз: при нисходящем методе парсер выполняет поиск сверху – с начального символа в коде и ищет подходящие ему синтаксические связи. Таким образом, древо синтаксического анализа разрастается сверху вниз, в направлении более детальной разбивки.
* Снизу-вверх: восходящий парсер начинает снизу, с самого нижнего символа строки, а затем устанавливает все более крупные синтаксические связи. Это делается до тех пор, пока не будет достигнут начальный символ кода.

## 2.2 Постановка задачи парсинга сайта Центрального Банка России

Необходимо реализовать парсер динамики курса валют с сайта Центрального Банка России, который сможет в автоматическом режиме получать соответствующую информацию для заданной пользователем валюты за промежуток времени, начинающийся от указанной пользователем даты и заканчивающейся текущей датой. Парсер должен проходить по строкам таблицы на необходимой странице сайта и собирать из них информацию о курсе валюты за заданный период времени.

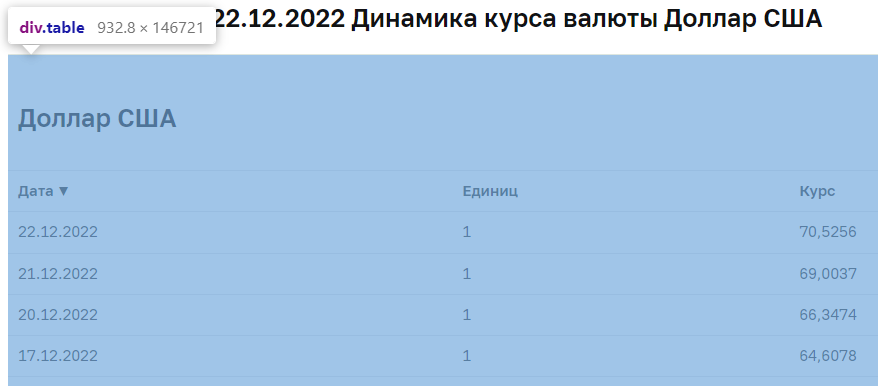
## 2.3 Сбор информации с сайта Центрального Банка России

На сайте Центрального Банка России есть раздел, на котором можно отслеживать динамику официального курса валют за заданный период времени (https://www.cbr.ru/currency\_base/dynamics/). Приведем пример url-адреса курса Доллара США за период от 01.01.2005 по 22.12.2022:

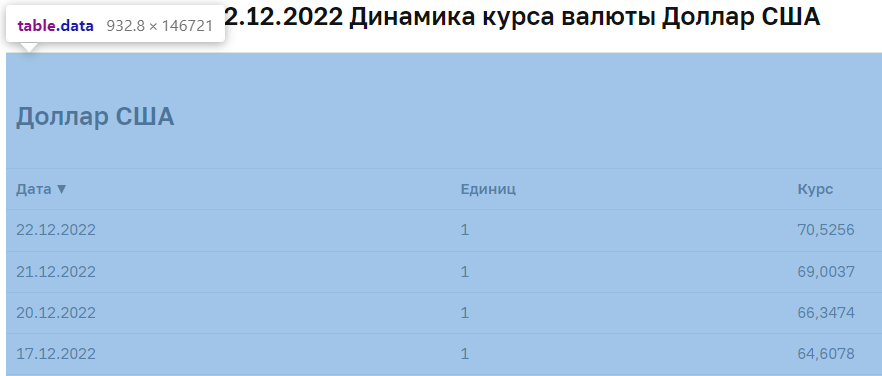
*https://www.cbr.ru/currency\_base/dynamics/?UniDbQuery.Posted=True&UniDbQuery.so=1&UniDbQuery.mode=1&UniDbQuery.date\_req1=&UniDbQuery.date\_req2=&UniDbQuery.VAL\_NM\_RQ=R01235&UniDbQuery.From=01.01.2005&UniDbQuery.To=22.12.2022*

Проанализировав этот URL-запрос, легко заметить, что в нем содержится информация о том, курс какой валюты нужно получить после запроса (в данном случае это число *1235*), а также даты, начиная с которой («*From=01.01.2005*») и по какую («*To=22.12.2022*») должна собираться информация.

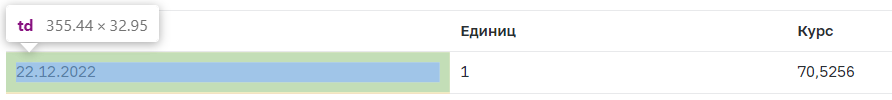
Просматривая сайт с заданной валютой, можно увидеть, что информация предоставляется в виде таблице с тремя колонками: Дата, Единиц и Курс. В коде страницы данная таблица находится в <div class="table">:



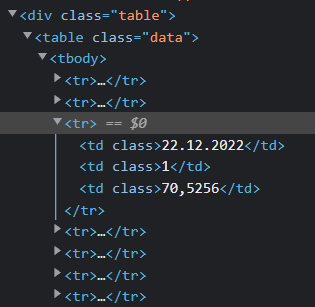
Данные в ней находятся в <table class="data">:



А каждая из ячеек находится в своей <td>:



В коде страницы это все представлено таким образом:



Знаю всю информацию о том, как формировать url-адрес и получать информацию с сайта, можно написать код программы, реализующий поставленную задачу.

# 3 Описание работы приложения

## 3.1 Описание разработанного функционала

Была поставлена задача реализовать программу, которая будет в аргументах командной строки получать информацию о том, динамику какой валюты нужно получить, дату начала парсинга, а также название и расширение файла для сохранения полученной информации. Программа должна обрабатывать получаемые аргументы командной строки, затем по этим данным формировать url-запрос к сайту Центрального Банка России, получать с него необходимую информацию о курсе валюты за заданный период времени, затем проверять ее и сохранять в необходимом для пользователя формате. Программа также на всех этапах записывает в консоль логи о текущем событии и возможных ошибках на каком-то из этапов.

## 3.2 Используемый язык и библиотеки

В качестве языка программирования для написания программы был выбран Python, так как этот язык предоставляет широкий набор библиотек для удобного написания парсера, а затем дальнейшего сохранения полученной информации. Также Python – объектно-ориентированный язык программирования, благодаря чему можно упростить написание кода, выделяя необходимые классы.

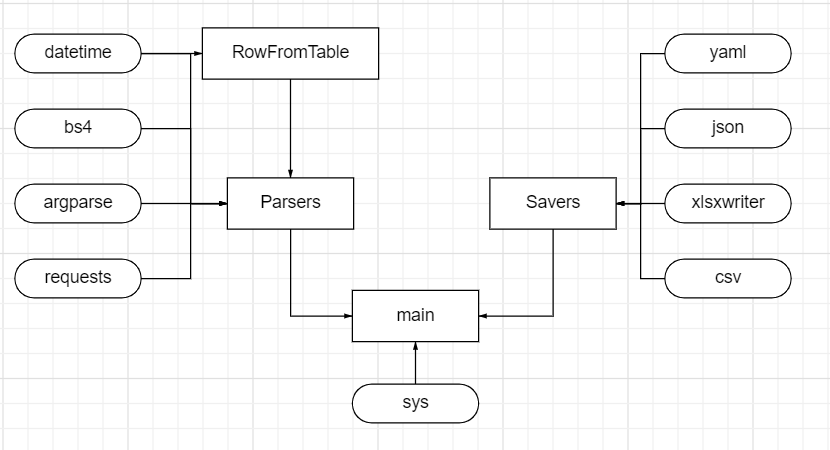
Для получения аргументов командной строки использована системная библиотека sys, а для их обработки используется библиотека argparse, которая позволяет разбирать аргументы, передаваемые программе при запуске из командной строки, и даёт возможность пользоваться этими аргументами в программе.

Для осуществления парсинга сайте Центрального Банка России используются библиотеки BeautifulSoup4 и requests. С помощью первой можно легко осуществлять поиск необходимой информации на странице, а вторая помогает извлекать найденную информацию. Использование связки этих библиотек существенно облегчает парсинг сайтов.

Для преобразования даты из строки в дату, понятную компьютеру используется системная библиотека datetime.

Для сохранения информации в необходимые файлы используются библиотеки yaml, json, xlsxwriter и csv. Эти библиотеки преобразуют и сохраняют полученные данные в необходимые пользователю расширения. Использование этих библиотек ускоряет сохранение файлов, так как не приходится обращаться к более сложным библиотекам, таким как pandas.

## 3.3 Архитектура приложения



В основе приложения находится файл main.py, который является входной точкой в приложение и объединением всех компонентов программы. В данном файле происходит основная логика приложения. В файле Parsers.py находится парсер аргументов командной строки, URLBuilder для составления url-адреса и парсер сайта Центрального Банка России. В файле RowFromTable.py находится структура RowFromTable, хранящая в себе информацию об одной из строчек таблицы на сайте ЦБ. Файл Savers.py содержит в себе родительский класс Writer и дочерние для него классы необходимые для сохранения информации. Выше приведена схема связей компонентов программы.

## 3.4 Алгоритм работы программы

Функция \_\_main\_\_ запускает приложение, обрабатывает полученные аргументы командной строки с помощью ArgParser, получает список структур RowFromTable с помощью функции Parse с аргументами начальной даты и валюты, а затем сохраняет полученную информацию с помощью SaveData с аргументами названия файла и расширения.

Сохранение полученного списка структур осуществляется в форматы yaml, txt, json, xlsx или csv по выбору пользователя. Для записи создается родительский класс с общим интерфейсом: инициализацией и записью данных в файл с заданным названием. На основе этого родительского класса создаются пять дочерних классов: YamlWriter, TxtWriter, JsonWriter, XlsxWriter и CsvWriter, каждый из которых обладает интерфейсом родительского класса и записывает информацию в файл с соответствующим расширением.

С применением данных классов реализована функция SaveData, принимающая список структур RowFromTable и название и расширение выходного файла, которая в зависимости от указанного пользователем расширения создает соответствующий writer и сохраняет данные. Функция проверяет необходимый формат файла, полученный от пользователя, чтобы он был одним из приведенных выше, а также отслеживает ошибки, которые могут возникнуть при записи файла.

## 3.5 Параметры запуска приложения

Для каждого из аргументов предусматриваются аргументы по умолчанию: стартовая дата по умолчанию - 01.01.2005, валюта по умолчанию - Доллар США, выходной файл по умолчанию называется как "ParseFile dd.mm.yyyy", с указанием текущей даты, формат для сохранения файла по умолчанию - csv.

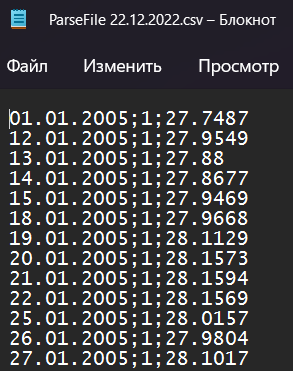
Программа принимает параметры парсинга и дальнейшего сохранения информации в виде аргументов командной строки. Программа обрабатывает следующие аргументы:

* --startDate=dd.mm.yyyy – дата начала парсинга сайта ЦБ, по умолчания это дата 01.01.2005;
* --currency=USD – валюта для парсинга, указывается в виде названия для USD или EUR (строчными или заглавными буквами) или в виде номера для этих или других валют. Валюта по умолчанию – Доллар США;
* --outputFile=’ParseFile dd.mm.yyyy’ – название получаемого файла. По умолчанию - ’ParseFile dd.mm.yyyy’ с указанием даты на момент запуска программы;
* --format=csv – расширение для файла получаемой информации. Возможные варианты: yaml, txt, json, xlsx или csv. По умолчанию файл сохраняется в расширении csv.

# 4 Проверка работоспособности приложения

Сделаем несколько тестовых запусков приложения и проверим корректность его работы.

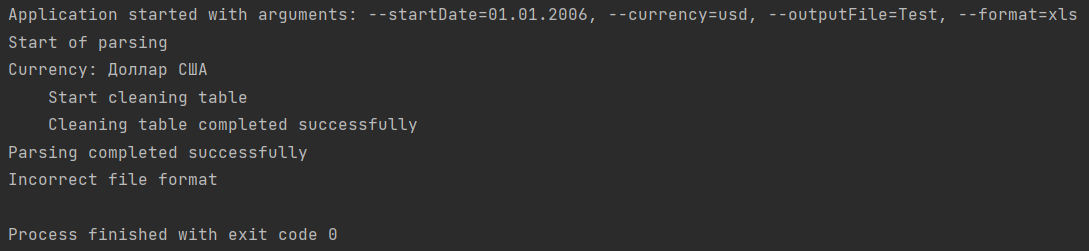
* Для начала запустим приложение без аргументов командной строки. В этом случае мы должны получить файл "ParseFile dd.mm.yyyy.csv" с информацией о курсе Доллара США за период с 01.01.2005 по текущую дату. В результате работы программы в консоли отображено, что файл сохранен успешно и мы получаем файл "ParseFile 22.12.2022" с содержимым вида:



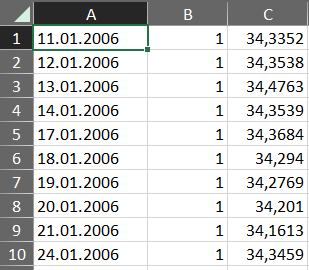
и так далее до текущей даты.

То есть на данном тестовом примере программа работает корректно.

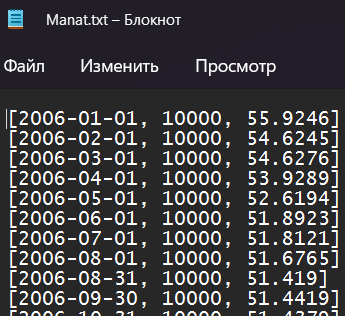
* Далее проверим работоспособность программы аналогично на Долларе США, но уже со стартовой датой от 01.01.2006 и сохранением в файл c названием "Test" и расширением xls, то есть проверим работоспособность программы с расширением, с которым программа не может сохранять файл. В результате в консоль выводится сообщение "Incorrect file format", то есть исключение обрабатывает корректно.



* Далее проверим работу при сохранении информации в формат xlsx данных о курсе евро начиная с 01.01.2006 в файл с названием "Test". В результате программа должна корректно выполниться и записать соответствующую информацию. В результате работы программы в консоли выводится сообщение “File saved successfully” о корректной записи файла, а в "Test.xlsx" таблица из 3-х колонок с датой, количеством и курсом за даты, начиная с 01.01.2006 (первая запись в таблице от 11.01.2006). Содержимое файла “Test.xlsx”:



* Далее проверим работу для валюты с номером 1710 - Туркменского маната. Этот тест будет проверять правильность работы программы, если в таблице присутствуют посторонние записи, так как сейчас она называется Туркменский манат, а до 2009 года название было просто Туркменский манат. Сохранять будем в файл "Manat" с расширением txt с получением информации от 01.01.2006. В результате работы программы парсинг проходит удачно и соответствующий файл успешно сохраняется. Содержимое полученного файла:



# 5 Заключение

В результате выполнения работы удалось написать приложение, которое получает информацию с сата Центрального Банка России о курсах валют за заданный период времени, проверяет данные, преобразует их в структуру RowFromTable, а затем записывает в необходимый файл полученную информацию. При работе программа выводит в консоль информацию о происходящих в данный момент событиях и успешности их выполнения, в случае возникновения ошибки программа останавливается и выводит информацию в консоль.

# 6 Список использованных источников

* Информация о работе парсера

https://romi.center/ru/learning/article/what-is-data-parsing

* Документация по библиотеке BeautifulSoup

https://www.crummy.com/software/BeautifulSoup/bs4/doc.ru/index.html

* Документация по библиотеке datatime

https://docs.python.org/3/library/datetime.html#module-datetime

* Документация по библиотеке argparse

https://docs.python.org/3/library/argparse.html

* Документация по библиотеке requests

https://requests.readthedocs.io/en/latest/

* Документация по библиотеке json

https://docs.python.org/3/library/json.html

* Документация по библиотеке xlsxwriter

https://xlsxwriter.readthedocs.io/

* Документация по библиотеке yaml

https://pyyaml.org/wiki/PyYAMLDocumentation

* Документация по библиотеке csv

<https://docs.python.org/3/library/csv.html>

# 7 Приложения

## 7.1 Приложение 1. Файл Parsers.py

from RowFromTable import RowFromTable

from datetime import date as dtDate

from datetime import datetime

from bs4 import BeautifulSoup

import requests

import argparse

class URLBuilder:

def \_\_init\_\_(self, currency, startDate):

currencyDict = {'usd': '1235', 'eur': '1239'}

currency = currency if currency.isdigit()\

else currencyDict[currency.lower()]

currentDate = datetime.now()

currentDate = f'{currentDate.day}.{currentDate.month}.{currentDate.year}'

self.\_\_url = 'https://www.cbr.ru/currency\_base/dynamics/'

self.\_\_url += '?UniDbQuery.Posted=True&UniDbQuery.so=1&UniDbQuery.mode=1'

self.\_\_url += '&UniDbQuery.date\_req1=&UniDbQuery.date\_req2=&UniDbQuery.VAL\_NM\_RQ=R0'

self.\_\_url += currency

self.\_\_url += '&UniDbQuery.From=' + startDate

self.\_\_url += '&UniDbQuery.To=' + currentDate

def GetURL(self):

return self.\_\_url

def ArgParser():

parser = argparse.ArgumentParser()

parser.add\_argument('--startDate', default='01.01.2005')

parser.add\_argument('--currency', default='USD')

parser.add\_argument('--outputFile', default=f'ParseFile {datetime.now().day}.{datetime.now().month}.{datetime.now().year}')

parser.add\_argument('--format', default='csv')

return parser

def Parse(startDate, currency):

print('Start of parsing')

try:

urlBuilder = URLBuilder(currency, startDate)

page = requests.get(urlBuilder.GetURL())

bs = BeautifulSoup(page.text, "lxml")

table = bs.find('table', 'data')

print('Currency:',

table.find('td').get\_text().replace('\n', ''))

data = []

rows = table.find\_all('td')

print('\tStart cleaning table')

clearRows = []

for row in rows:

try:

date = row.get\_text().split('.')

date.reverse()

dtDate.fromisoformat('-'.join(date))

clearRows.append(row)

continue

except Exception:

pass

try:

int(row.get\_text())

clearRows.append(row)

continue

except Exception:

pass

try:

float(row.get\_text().replace(',', '.'))

clearRows.append(row)

continue

except Exception:

pass

rows = clearRows

print('\tCleaning table completed successfully')

for i in range(-1, -len(rows), -3):

data.append(RowFromTable(rows[i-2], rows[i-1], rows[i]))

except Exception:

print("Failed to parse site")

exit()

else:

print('Parsing completed successfully')

return data

## 7.2 Приложение 2. Файл RowFromTable.py

from datetime import date as dtDate

class RowFromTable:

def \_\_init\_\_(self, date, count, curs):

date = date.get\_text().split('.')

date.reverse()

self.\_\_date = dtDate.fromisoformat('-'.join(date))

self.\_\_count = int(count.get\_text())

self.\_\_curs = float(curs.get\_text().replace(',', '.'))

def GetData(self):

return [self.\_\_date, self.\_\_count, self.\_\_curs]

## 7.3 Приложение 3. Файл Savers.py

import yaml

import json

import xlsxwriter

import csv

class Writer:

def \_\_init\_\_(self, param):

print(f'Initialization of {param} writer')

def WriteData(self, outputFile):

print(f'Write data to {outputFile} file')

class YamlWriter(Writer):

def \_\_init\_\_(self, data):

super().\_\_init\_\_('yaml')

self.\_\_data = data

def WriteData(self, outputFile):

super().WriteData(outputFile + '.yaml')

data = []

for line in self.\_\_data:

data.append(line.GetData())

with open(outputFile + '.yaml', 'w') as file:

yaml.dump(data, file)

class TxtWriter(Writer):

def \_\_init\_\_(self, data):

super().\_\_init\_\_('txt')

self.\_\_data = data

def WriteData(self, outputFile):

super().WriteData(outputFile + '.txt')

with open(outputFile + '.txt', 'w') as file:

for line in self.\_\_data:

file.write('[' + ', '.join(map(str, line.GetData()) + ']\n')

class JsonWriter(Writer):

def \_\_init\_\_(self, data):

super().\_\_init\_\_('json')

self.\_\_data = data

def WriteData(self, outputFile):

super().WriteData(outputFile + '.json')

data = []

for i in range(len(self.\_\_data)):

x = self.\_\_data[i].GetData()

date = x[0].isoformat().split('-')

date.reverse()

x[0] = '.'.join(date)

data.append(json.dumps({i: x}))

data = ', '.join(data)

with open(outputFile + '.json', 'w') as file:

file.write(data)

class XlsxWriter(Writer):

def \_\_init\_\_(self, data):

super().\_\_init\_\_('xlsx')

self.\_\_data = data

def WriteData(self, outputFile):

super().WriteData(outputFile + '.xlsx')

data = []

for i in range(len(self.\_\_data)):

x = self.\_\_data[i].GetData()

date = x[0].isoformat().split('-')

date.reverse()

x[0] = '.'.join(date)

data.append(x)

workbook = xlsxwriter.Workbook(outputFile + '.xlsx')

worksheet = workbook.add\_worksheet()

for i in range(len(self.\_\_data)):

worksheet.write\_row(i, 0, data[i])

workbook.close()

class CsvWriter(Writer):

def \_\_init\_\_(self, data):

super().\_\_init\_\_('csv')

self.\_\_data = data

def WriteData(self, outputFile):

super().WriteData(outputFile + '.csv')

data = []

for i in range(len(self.\_\_data)):

x = self.\_\_data[i].GetData()

date = x[0].isoformat().split('-')

date.reverse()

x[0] = '.'.join(date)

data.append(x)

with open(outputFile + '.csv', 'w') as file:

writer = csv.writer(file, delimiter=';')

writer.writerows(data)

def SaveData(data, outputFile, formatFile):

try:

if formatFile.lower() == 'yaml':

writer = YamlWriter(data)

elif formatFile.lower() == 'txt':

writer = TxtWriter(data)

elif formatFile.lower() == 'json':

writer = JsonWriter(data)

elif formatFile.lower() == 'xlsx':

writer = XlsxWriter(data)

elif formatFile.lower() == 'csv':

writer = CsvWriter(data)

else:

print('Incorrect file format')

exit()

writer.WriteData(outputFile)

except Exception:

print('Failed to save file')

exit()

else:

print('File saved successfully')

## 7.4 Приложение 4. Файл main.py

import sys

from Parsers import ArgParser, Parse

from Savers import SaveData

if \_\_name\_\_ == '\_\_main\_\_':

print('Application started with arguments:',

str(sys.argv[1:]).replace('[', '').\

replace(']', '').replace("'", ''))

argParser = ArgParser()

arguments = argParser.parse\_args(sys.argv[1:])

data = Parse(arguments.startDate, arguments.currency)

SaveData(data, arguments.outputFile, arguments.format)