Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ст. преподаватель |  |  |  | М. Д. Поляк |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ |
| Парсинг таблиц в MS Word |
| по дисциплине: Методы трансляций |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4831 |  |  |  | К. А. Корнющенков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

Оглавление

[Введение 3](#_Toc59377921)

[Постановка задачи 3](#_Toc59377922)

[Структура разработанной программы 4](#_Toc59377923)

[Результаты тестирования 4](#_Toc59377924)

[Руководство пользователя 5](#_Toc59377925)

[Заключение 6](#_Toc59377926)

[Библиографический список 6](#_Toc59377927)

[Приложение 7](#_Toc59377928)

# Введение

В современном мире, где всем управляют информационные технологии ручная обработка данных теряет смысл.

Именно поэтому необходимо автоматизировать такие процессы, как получение и обработку больших текстовых файлов, таблиц и других различных документов, которые имею общую структуру, которую можно подогнать под единых шаблон, чтобы сэкономить время и ресурсы на этапе обработки исходной информации.

В ходе выполнения курсового проекта был написан программный модуль, которое позволяет быстро и эффективно получать информацию о фитопланктоне.

Создание подобного модуля является довольно актуальной задачей, решение которой позволит значительно повысить эффективность работы аналитиков.

# Постановка задачи

В ходе выполнения курсового проекта необходимо написать программу на высокоуровневом языке программирования общего названия Python для обработки таблиц из файлов Microsorf Word в файлы текстового формата с расширением TSV. Для реализации данной задачи недостаточно стандартных возможностей выбранного нами языка программирования именно поэтому нами были выбраны следующие библиотеки:

* Csv – данный модуль дает программисту возможность выполнять структурный анализ файлов CSV, который будет необходим нам для записи обработанных данных в текстовый файл.
* docx2python – данный модуль дает программисту возможность работать с текстовыми файлами Docx, который будет необходим нам для считывания исходных данных.
* Enum – данный модель дает программисту возможность ограничить множество допустимых значений для некоторых типов данных. Использовать данную возможность мы будем для того, чтобы понимать какую таблицу программа обрабатывает в тот или иной момент времени.

# Структура разработанной программы

PhytoplanktonEnum - перечисление, которое служит для отслеживания какая часть таблицы о фитопланктоне обрабатывается в данный момент.

Данное перечисление содержит 4 поля:

* header – используется, когда парсится первый блок таблицы
* body – используется, когда парсится второй блок таблицы
* bottom – используется, когда парсится третий блок таблицы
* notGood – используется, когда таблица не начала парсится

HeaderModel – модель, которая предназначена для хранения данных первого блока таблицы. Хранит в себе такие поля, как название водоема, дата замеров, номер станции, глубина водоема, температура воду, прозрачность воды, автор данных

TaksonOrDepartamentModel – модель, которая предназначена для хранения данных второго и третьего блока таблицы. Хранит в себе такие поля, как название таксона или отдела, численность, биомасса, численность в процентном соотношении, биомасса в процентном соотношении

PhytoplanktonModel – модель, которая предназначена для хранения данных всех таблицы. Хранит в себе экземпляр HeaderModel и 2 экземпляра TaksonOrDepartamentModel

ParsePhytoplankton – класс, который считывает, обрабатывает и записывает исходные данные в нужный выходной формат. Содержит в себе такие методы, как headerParse, taksonOrDepartamentParse, saveData, startParse.

# Результаты тестирования

Тестирование класса ParsePhytoplankton

|  |  |  |  |
| --- | --- | --- | --- |
| Имя теста | Описание сценария | Входные данные | Выходные данные |
| dataTest | Создается экземпляр класса ParsePhytoplankton. В качестве входного параметра передается имя файла с его расширением. | data.docx – файл, который находится в корневой папке проекта и содержит только необходимую информацию для парсинга | 3 файла, которые автоматически создались и заполнились данными, которые находились в data.docx |
| dataTest2 | Создается экземпляр класса ParsePhytoplankton. В качестве входного параметра передается имя файла с его расширением. | data.docx – файл, который находится в корневой папке проекта и содержит только необходимую информацию и лишнюю информацию для парсинга | 3 файла, которые автоматически создались и заполнились данными, которые находились в data.docx |
| dataTest3 | Создается экземпляр класса ParsePhytoplankton. В качестве входного параметра передается имя файла с его расширением. | data.docx – файл, который находится в корневой папке проекта и содержит только лишнюю информацию для парсинга | 3 пустых файла |
| dataTest4 | Создается экземпляр класса ParsePhytoplankton. В качестве входного параметра передается имя файла с его расширением. | data1.docx – файл, который не существует | Сообщение “Файл не найден” |

# Руководство пользователя

# Для того, чтобы воспользоваться данной программой необходимо создать экземпляр класса ParsePhytoplankton и в качестве входного параметра передать путь к файлу с исходными данными. Данный файл должен иметь расширение DOCX и находиться в корневой папке проекта.

# Пример вызова программы: ParsePhytoplankton(“data.docx”)

Если программа не сможет обнаружить файл с данными, то в консоле будет выведено сообщение “Файл не найден”

Если файл обнаружен, то программа считает исходные данные, обработает их и запишет результат в 3 файла, которые будут располагаться в корневой директории проекта.

Файл headerPhytoplankton.tsv будет содержать данные о заголовках всех таблиц

Файл bodyPhytoplankton.tsv будет содержать данные о таксоне всех таблиц

Файл bottonPhytoplankton.tsv будет содержать данные отделе всех таблиц

Пример работы программы можно посмотреть в файле README.md в репозитории, ссылк на который представлена в разделе “Биографический список”

# Заключение

В ходе курсового проекта было разработан программный модуль, который позволяет быстро и эффективно парсить таблицы. Также данный модуль соответствует сформулированному техническому заданию.

К достоинствам приложения можно отнести простоту и наглядность в использовании.

К недостаткам слабая переиспользуемость компонентов и достаточно медленную работу при больших исходных данных.

Результатом выполнения курсового проекта стало:

- закрепление знаний, полученных в ходе изучения дисциплины «Методы трансляций»

- приобретение навыков практического программирования с использованием библиотек: docx2python и сsv

# Библиографический список

* Корнющенков К. А. Программа для парсинга таблиц из файла с расширением .docx URL: <https://github.com/Kiryakor/parsePhytoplankton>
* Официальная документация docx2python URL: <https://pypi.org/project/docx2python/>
* Официальная документация python URL: https://www.python.org/doc/

# Приложение

from docx2python import docx2python

from enum import Enum

import csv

class PhytoplanktonEnum(Enum):

"""Перечисление для отслеживания таблицы, которая парсится в данный момент"""

header = 1

body = 2

bottom = 3

notGood = 4

class HeaderModel:

"""Модель для хранения свойств заголовка таблицы"""

def \_\_init\_\_(self, water="", date="", station="", depth="", temperature="", alpha="", author=""):

self.water = water

self.date = date

self.station = station

self.depth = depth #глубина

self.temperature = temperature

self.alpha = alpha

self.author = author

class TaksonOrDepartmentModel:

"""Модель для хранения свойств таблиц: Отдел и Таксон"""

def \_\_init\_\_(self, takson, counter, bioMassa, percentCounter, percentBioMassa):

self.takson = takson

self.counter = counter

self.bioMassa = bioMassa

self.percentCounter = percentCounter

self.percentBioMassa = percentBioMassa

class PhytoplanktonModel:

"""Модель для ханения данных одной полной таблице о фитопланктоне"""

def \_\_init\_\_(self, head, body, bottom):

self.head = head

self.body = body

self.bottom = bottom

class ParsePhytoplankton:

"""Класс для парсинга данных о фитопланктоне"""

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

""" path - путь к файлу с исходными данными """

self.path = path

self.parseData = []

self.parseState = PhytoplanktonEnum.notGood

self.isStart = True

try:

self.doc\_result = docx2python(path)

except:

print("Файл не найден")

isStart = False

if self.isStart:

self.startParse()

def headerParse(self, content, header):

"""Метод для парсинг заголовка таблицы"""

if "Водоем:" in content:

frst = content.find("Дата")

scnd = content.find("Станция")

header.water = content[7:frst].strip()

header.date = content[frst + 5:scnd].strip()

header.station = content[scnd + 8:].strip()

elif "Глубина:" in content:

frst = content.find("Температура")

scnd = content.find("Прозрачность")

header.depth = content[8:frst].strip()

header.temperature = content[frst + 12:scnd].strip()

header.alpha = content[scnd + 13:].strip()

elif "Исполнитель:" in content:

header.author = content[12:].strip()

def taksonOrDepartmentParse(self, content):

"""Метод для парсинг таблиц: Отдел и Таксон"""

if content[0][0] != "Отдел" and content[0][0] != "Таксон":

return TaksonOrDepartmentModel(content[0][0], content[1][0], content[2][0], content[3][0], content[4][0])

return 0

def saveData(self):

"""

Метод для сохранения данных в файлы:

headerPhytoplankton.tsv,

bodyPhytoplankton.tsv,

bottomPhytoplankton.tsv

"""

with open('headerPhytoplankton.tsv', 'wt') as out\_file:

tsv\_writer = csv.writer(out\_file, delimiter='\t')

counter = 0

for i in self.parseData:

tsv\_writer.writerow(

[counter, i.head.water, i.head.date, i.head.station, i.head.depth, i.head.temperature, i.head.alpha,

i.head.author])

counter += 1

with open('bodyPhytoplankton.tsv', 'wt') as out\_file:

tsv\_writer = csv.writer(out\_file, delimiter='\t')

counter = 0

for i in self.parseData:

for j in i.body:

tsv\_writer.writerow([counter, j.takson, j.counter, j.bioMassa, j.percentCounter, j.percentBioMassa])

counter += 1

with open('bottomPhytoplankton.tsv', 'wt') as out\_file:

tsv\_writer = csv.writer(out\_file, delimiter='\t')

counter = 0

for i in self.parseData:

for j in i.bottom:

tsv\_writer.writerow([counter, j.takson, j.counter, j.bioMassa, j.percentCounter, j.percentBioMassa])

counter += 1

def startParse(self):

"""Главный метод для парсинга"""

for j in self.doc\_result.body:

header = HeaderModel()

bottom = []

body = []

for i in j:

if i[0][0] == "":

continue

if "Водоем:" in i[0][0]:

self.parseState = PhytoplanktonEnum.header

elif "Таксон" == i[0][0]:

self.parseState = PhytoplanktonEnum.body

elif "Отдел" == i[0][0]:

self.parseState = PhytoplanktonEnum.bottom

elif "Всего" == i[0][0]:

takson = self.taksonOrDepartmentParse(i)

if takson != 0:

bottom.append(takson)

self.parseState = PhytoplanktonEnum.notGood

if self.parseState == PhytoplanktonEnum.header:

self.headerParse(i[0][0], header)

elif self.parseState == PhytoplanktonEnum.body or self.parseState == PhytoplanktonEnum.bottom:

takson = self.taksonOrDepartmentParse(i)

if takson != 0 and self.parseState == PhytoplanktonEnum.body:

body.append(takson)

elif takson != 0 and self.parseState == PhytoplanktonEnum.bottom:

bottom.append(takson)

if header.author != "":

self.parseData.append(PhytoplanktonModel(header, body, bottom))

self.saveData()

ParsePhytoplankton('data.docx')