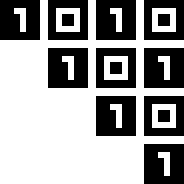
Министерство науки и высшего образования РФ

ФГБОУ ВО «Тамбовский государственный технический университет»

Кафедра

«САПР»



Отчет по лабораторной работе №1

по дисциплине «Алгоритмизация, модели данных и основы ИИ»

Выполнил: студент группы БВТ241

Белов А.К.

Проверил: преподаватель

Патутин К. И.

Тамбов 2025

**Цель**  
Создать программу по заполнению шаблонных документов  
  
 **Задание 1**  
Имеется архив расписания занятий ИАиИТ с 2017-го по 2023-й годы. Необходимо написать программу, которая сравнивает две версии расписания за один и тот же семестр и находит список различий между ними.  
  
 **Решение**  
Используя библиотеку docx считываю данные, сохраняю стиль а после заполняю с приминением стиля на новый текст  
 **Листинг:  
  
Из файла ...\_Graphics\laba\_logging.py.cpp**import os  
import time  
  
LOG\_FILE = 'loggs.txt'   
inited = []  
def InitLogFile():  
 with open(LOG\_FILE, "w", encoding="utf-8") as f:  
 # fcntl.flock(f, fcntl.LOCK\_EX) # Блокировка файла  
 f.write("=== Start logging ===\n")  
 InitFile(\_\_file\_\_)  
 # fcntl.flock(f, fcntl.LOCK\_UN) # Разблокировка  
  
  
def InitFile(filename):  
 with open(LOG\_FILE, "a+", encoding="utf-8") as f:  
 print("add - "+filename)  
 if filename not in inited:  
 f.write("-init|" + str(filename) + "\n")   
 inited.append(filename)  
 time.sleep(0.1)  
  **Из файла ...5\_Graphics\replacement.py.cpp**from docx import Document  
from docx.oxml.ns import qn  
from docx.oxml import parse\_xml  
import re  
  
from laba\_logging import \*  
  
InitFile(\_\_file\_\_)  
  
def replace\_in\_paragraph(paragraph, old\_text, new\_text):  
 if old\_text in paragraph.text:  
  
 style = paragraph.style  
 runs = paragraph.runs  
   
 full\_text = ''.join(run.text for run in runs)  
 print(full\_text)  
  
 if old\_text in full\_text:  
 paragraph.clear()  
 parts = full\_text.split(old\_text)  
   
 for i, part in enumerate(parts):  
 if part:   
 paragraph.add\_run(part)  
   
 if i < len(parts) - 1:  
 run = paragraph.add\_run(new\_text)  
 if runs:  
 first\_run = runs[0]  
 run.bold = first\_run.bold  
 run.italic = first\_run.italic  
 run.underline = first\_run.underline  
 if first\_run.font.color:  
 run.font.color.rgb = first\_run.font.color.rgb  
 if first\_run.font.name:  
 run.font.name = first\_run.font.name  
 if first\_run.font.size:  
 run.font.size = first\_run.font.size  
 paragraph.style = style  
  
def replace\_in\_table(table, old\_text, new\_text):  
 for row in table.rows:  
 for cell in row.cells:  
 for paragraph in cell.paragraphs:  
 replace\_in\_paragraph(paragraph, old\_text, new\_text)  
 for nested\_table in cell.tables:  
 replace\_in\_table(nested\_table, old\_text, new\_text)  
  
  
def replace\_text\_in\_docx(doc, old\_text, new\_text):  
  
 # doc = Document(input\_file)  
  
 for paragraph in doc.paragraphs:  
 replace\_in\_paragraph(paragraph, old\_text, new\_text)  
   
 for table in doc.tables:  
 replace\_in\_table(table, old\_text, new\_text)  
   
   
 # doc.save(output\_file)  
  
  
def replace(docx, dict\_replacements):  
 for key in dict\_replacements:  
 replace\_text\_in\_docx(docx, key, dict\_replacements[key])  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 doc = Document("Реферат.docx")  
 replace\_text\_in\_docx(doc, "\_\_authors\_\_", "фувертпвот")  
 doc.save("output\_file.docx")  
   
  
  
# Для чего нужен модуль. Модуль используется для...  
# Что использует модуль для работы   
# Область применения данного модуля  
  
# \_\_authors\_\_  
# \_\_director\_\_  
# \_\_program\_\_  
# \_\_annotation\_\_  
# \_\_type\_\_  
# \_\_language\_\_  
# \_\_os\_\_  
# \_\_memory\_\_  
  
 **Из файла ...\_5\_Graphics\task1\main.py.cpp**import sys  
import os  
from docx import Document  
  
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))))  
from laba\_logging import \*  
InitLogFile()  
  
from replacement import \*  
  
InitFile(\_\_file\_\_)  
  
def err():  
 print("Вы что-то сделали неверно, попробуйте еще раз")  
 return input()  
  
  
def Referat():  
 print("Подготовка к созданию...")  
 doc = Document("D:\\projects\\VisualStudioCode\\Laba\_2\_2\_5\_Graphics\\templates\\Реферат.docx")  
 dict\_data = {  
 '\_\_authors\_\_' : "me"  
 }  
 print("Введите авторов через запятую: ")  
 tmp = input().replace(",", ",\n")  
 while len(tmp.split(".")) < 3:  
 tmp = err().replace(",", ",\n")  
 dict\_data['\_\_authors\_\_'] = tmp  
  
  
 print("Ведите правообладателя (ФИО) или Название организации: ")  
 dict\_data['\_\_director\_\_'] = input()  
  
 print("Ведите название вашего модуля: ")  
 dict\_data['\_\_program\_\_'] = input()  
 name = f"Реферат {dict\_data['\_\_program\_\_']}.docx"  
  
 print("Придумайте аннотацию к вашему модулю: ")  
 print("\t- Для чего нужен модуль. \n\t- Что использует модуль для работы \n\t- Область применения данного модуля")  
 tmp = input()  
 while (tmp.find("модул") < 0) or (tmp.find("для") < 0) or (len(tmp) < 15):  
 tmp = err()  
 dict\_data['\_\_annotation\_\_'] = tmp  
  
 print("Укажите тип ЭВМ: ")  
 dict\_data['\_\_type\_\_'] = input()  
  
 print("Укажите язык создания: ")  
 tmp = input()  
 while True:  
 f = True  
 for tmp\_ in tmp.split(", "):  
 if tmp\_ not in ['C++', "python", "java", 'питон', "C#", "C", ]:  
 tmp = err()  
 f = False  
 break  
 if f == True:  
 break  
 dict\_data['\_\_language\_\_'] = tmp  
   
 print("Укажите операционную систему: ")  
 dict\_data['\_\_os\_\_'] = input()  
   
 print("Укажите затрачиваемое количество памяти (в Мб): ")  
 dict\_data['\_\_memory\_\_'] = input()  
 print("Сохраняется... ")  
  
 replace(doc, dict\_data)  
 doc.save(name)  
 print(f"Сохранено {name}! ")  
  
  
  
  
def ConsentToInformation():  
 '''  
Белов Александр Кириллович  
22.04.2007  
admin  
Россия, Совхозная, 39а  
Русское  
0  
ксорти  
50  
'''  
 print("Подготовка к созаднию...")  
 doc = Document("D:\\projects\\VisualStudioCode\\Laba\_2\_2\_5\_Graphics\\templates\\Согласие на указание сведений об авторе.docx")  
 this\_dict = {  
 '\_\_name\_\_' : "program"  
 }  
  
 print("Введите ФИО полностью: ")  
 this\_dict['\_\_full\_user\_name\_\_'] = input()  
 user\_name = this\_dict['\_\_full\_user\_name\_\_'].split(" ")  
 for i in range(0, len(user\_name)):  
 if i == 0:  
 this\_dict['\_\_user\_name\_\_'] = user\_name[i] + " "  
 else:  
 this\_dict['\_\_user\_name\_\_'] += user\_name[i][0] + ". "  
   
  
 print("Введите дату рождения DD.MM.YYYY: ")  
 date = input().split(".")  
 while True:  
 try:  
 this\_dict['\_\_day\_\_'] = str(int(date[0]))  
 this\_dict['\_\_month\_\_'] = str(int(date[1]))  
 this\_dict['\_\_year\_\_'] = str(int(date[2]))  
 break  
 except:  
 print("что то пошло не так, попробуйте еще")  
  
 print("Введите свою должность: ")  
 this\_dict['\_\_post\_\_'] = input()  
  
 print("Введите место постоянного жительства, включая указание страны: ")  
 this\_dict['\_\_residence\_\_'] = input()   
   
 print("Введите гражданство: ")  
 this\_dict['\_\_nationality\_\_'] = input()  
   
 print("Выбрать что-то одно из нижеприведенного: \n0 - если Физ. Лицо \n1 - если Юр. Лицо")  
 while(1):  
 a = input()  
 if a == "0":  
 print("секунду ... ")  
 this\_dict['\_\_info\_\_'] = this\_dict['\_\_user\_name\_\_'] + " " + this\_dict['\_\_residence\_\_']  
 break  
 if a == "1":  
 print("Введите наименование, место нахождения, основной государственный регистрационный номер (ОГРН) и идентификационный номер налогоплательщика (ИНН)")  
 this\_dict['\_\_info\_\_'] = input()  
 break  
 else:  
 print("что то не так, попробуйте еще раз")  
  
   
   
 print("Введите название проекта: ")  
 this\_dict['\_\_name\_\_'] = input()  
  
 print("Укажите сколько процентов работы вы выполняли: ")  
 percent = input()  
 while True:  
 try:  
 this\_dict['\_\_percent\_\_'] = str(int(percent)%101)  
 break  
 except:  
 print("что то пошло не так, попробуйте еще")  
  
 # print("Укажите сколько процентов работы вы выполняли: ")  
 # this\_dict['\_\_residence\_\_'] = input()  
  
 replace(doc, this\_dict)  
  
 doc.save("Согласие на указание сведений об " + this\_dict['\_\_full\_user\_name\_\_'] + ".docx")  
  
  
ConsentToInformation()  
# Referat()  
# A.K.<tkjd^W  
# sghfggfgn  
# zdsfbhdgnfgh  
# модуль нужен для тестирования  
# ывпв  
# C  
# zdfgdf  
# 10  
'''  
Белов Алексаендр Кириллович  
22.04.2007  
разработчик  
россия д5  
российское  
0  
бот  
20  
'''  
  
  
  
 Ссылка https://github.com/albel-t/Laba\_2\_1\_5\_Graphics.git на главный репозиторий  
 **Отчет сгенерирован report.py:**   
from docx import Document  
from docx.shared import Pt, RGBColor  
from docx.enum.text import WD\_UNDERLINE  
from docx.shared import Inches, Cm, Pt  
import glob  
import os  
  
def addCode(name, paragraph):  
 p = paragraph.add\_run(name)  
 p.font.name = 'Consolas'  
 p.font.size = Pt(12)  
 return p  
  
def addTask(task, paragraph):  
 p = paragraph.add\_run(task)  
 p.font.name = 'Times\_New\_Roman'  
 p.font.size = Pt(14)  
 return p  
   
def addCodeTitle(code, paragraph):  
 p = addTask(code, paragraph)  
 p.font.bold = True  
 return p  
  
def addTaskTitle(text, paragraph):  
 p = addTask(text, paragraph)  
 p.font.bold = True  
 return p  
  
def addImage(path, image, paragraph):  
 p = addTask(f'  
{image}:' , paragraph)  
 p.add\_picture(path + '\' + image, width=Inches(4.0))  
 return p  
  
#visual studio  
# path\_to\_project = input()  
   
LOG\_FILE = 'loggs.txt'   
  
path\_to\_this\_document = os.path.dirname(os.path.abspath(\_\_file\_\_))  
  
path\_to\_template\_document = "D:\projects\Templates\шаблон отчета Кафедра САПР.docx"  
path\_to\_README = "README.md"  
path\_to\_imgs = path\_to\_this\_document + '\img'  
name\_laba = "laba\_1"  
  
doc = Document(path\_to\_template\_document)  
  
style = doc.styles['Normal']  
style.font.name = 'Times\_New\_Roman'  
style.font.size = Pt(14)  
p = doc.add\_paragraph(" ")  
  
file\_log = open(LOG\_FILE, "r", encoding="utf8").readlines()  
files = []  
for string in file\_log:  
 string = string.split('|')  
 if "-init" in string[0]:  
 files.append(string[1].replace('  
', ''))  
 print("добавлени путь:" + string[1])  
  
  
file\_README = open(path\_to\_README, "r", encoding="utf8").read()  
for paragraph\_README in file\_README.split("##"):  
 times\_README = paragraph\_README.split(":")  
 print(f"  
{times\_README}  
")  
 if len(times\_README) > 1:  
 addTaskTitle(f"  
{times\_README[0]}  
", p)  
 print(f"  
{times\_README[1]}  
")  
 addTask(times\_README[1] + "  
", p)  
  
  
addTaskTitle("  
Листинг:  
", p)  
  
for file\_name in files:  
 print("записан путь:" + file\_name)  
 addCodeTitle("  
Из файла ..." + file\_name[-25:] + '.cpp  
', p)  
 addCode(open(file\_name, "r", encoding="utf8").read(), p)  
  
for item in os.listdir(path\_to\_imgs):  
 print("добавлени путь:" + item)  
 addImage(path\_to\_imgs, item, p)  
  
doc.save(path\_to\_this\_document + '/Oтчет\_'+name\_laba+'.docx')