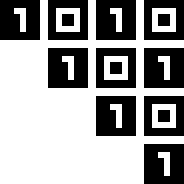
Министерство науки и высшего образования РФ

ФГБОУ ВО «Тамбовский государственный технический университет»

Кафедра

«САПР»



Отчет по лабораторной работе №1

по дисциплине «Алгоритмизация, модели данных и основы ИИ»

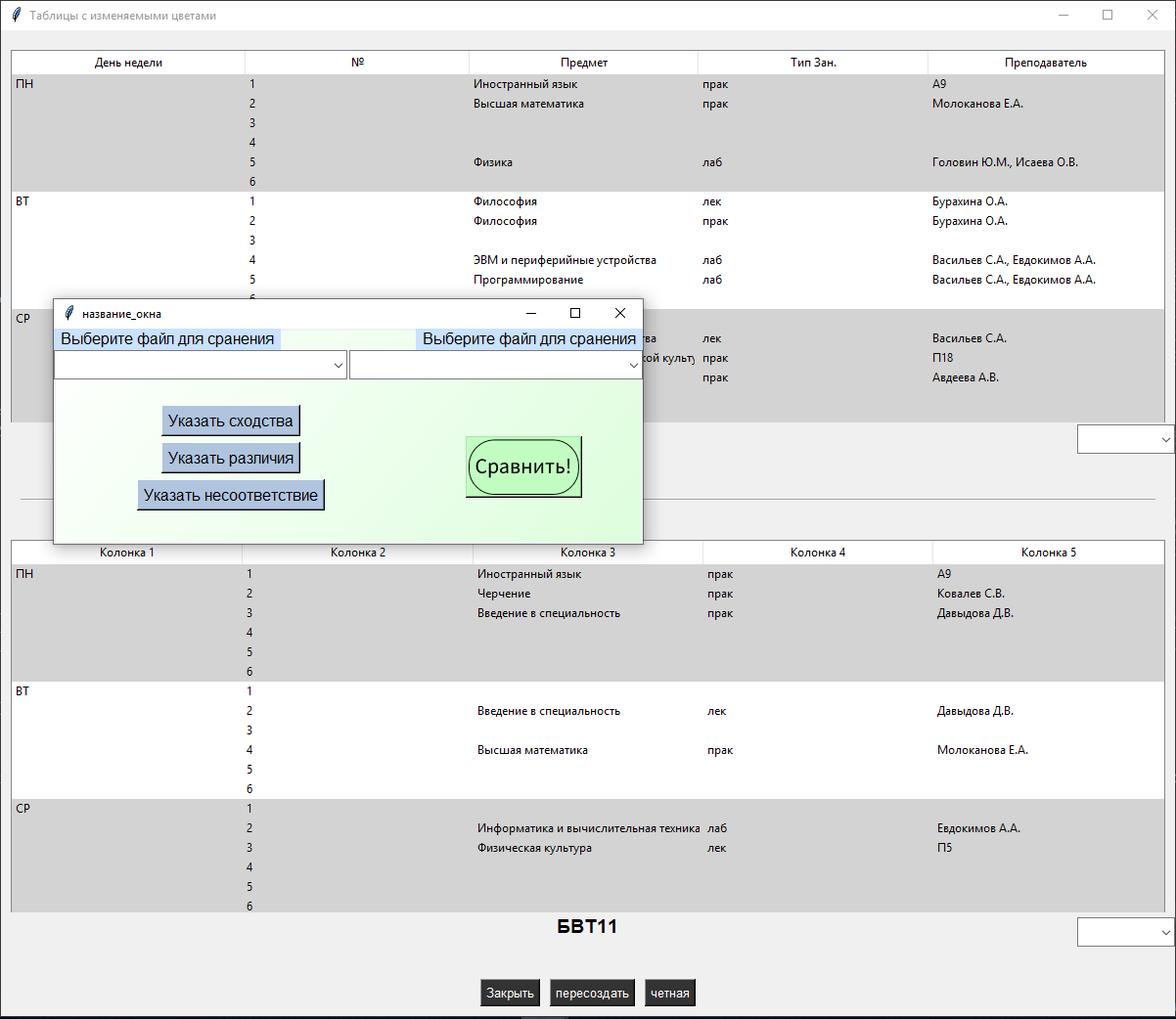
Выполнил: студент группы БВТ241

Белов А.К.

Проверил: преподаватель

Патутин К. И.

Тамбов 2025

**Цель**  
Создать приложение для сравнения распсаний имея таблицы данных   
  
 **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 xlrd считываю данные, сохраняю стиль а после заполняю таблицу с пометками в оккне созданом на основе tkinter  
 **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 ...ics\PythonApplication1.py.cpp**﻿import tkinter as tk  
from PIL import ImageTk, Image  
import os  
import sys  
  
from tkinter import ttk  
  
import random  
  
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))))  
from laba\_logging import \*  
InitLogFile()  
  
  
InitFile(\_\_file\_\_)  
  
from main import \*  
  
  
'''  
 # Настраиваем заголовки столбцов для второй таблицы  
 table2.heading('col1', text='№')  
 table2.heading('col2', text='Предмет')  
 table2.heading('col3', text='Тип Зан.')  
 table2.heading('col4', text='Преподаватель')  
 table2.heading('col5', text='Аудитория')  
   
 # Настраиваем ширину столбцов  
 table2.column('col1', width=10)  
 table2.column('col2', width=120)  
 table2.column('col3', width=10)  
 table2.column('col4', width=120)  
 table2.column('col5', width=40)  
'''  
  
  
  
  
class TableWindow:  
 def \_\_init\_\_(self, new\_window, table1, table2):  
 self.new\_window = new\_window  
 self.new\_window.title("Таблицы с изменяемыми цветами")  
 self.new\_window.geometry("1200x700")  
 self.new\_window.minsize(1200, 700)  
 self.groupid1 = 0  
 self.groupid2 = 0  
 # Цветовые палитры  
 # self.colors1 = ['#FF0000', '#00FF00', '0000FF']  
 self.label1 = None  
 self.frame1 = None  
 self.table1 = None  
  
 self.separator = None  
   
 self.label2 = None  
 self.frame2 = None  
 self.table2 = None  
   
 self.listtable1 = table1  
 self.listtable2 = table2  
  
 self.parity = 0  
  
  
  
 # Создаем таблицы  
 self.create\_tables()  
   
  
  
 # Создаем кнопки для изменения цветов  
 self.create\_buttons(table1, table2)  
   
 def create\_tables(self): #, listtable1: list[schedule], listtable2: list[schedule]  
 # # Создаем фрейм для первой таблицы  
 # if self.current\_table1 != None:  
   
 # if self.current\_table2 != None:  
  
 table1data = self.listtable1[self.groupid1]  
 table2data = self.listtable2[self.groupid2]  
 if self.frame1 == None:  
 self.frame1 = tk.Frame(self.new\_window)  
 self.frame1.pack(pady=20, padx=10, fill='both', expand=True)  
   
   
 # Первая таблица  
 if self.table1 == None:  
 self.table1 = ttk.Treeview(self.frame1, columns=('col1', 'col2', 'col3', 'col4', 'col5'), show='headings', height=12)  
   
 # Настраиваем заголовки столбцов  
 col\_name = ["День недели", "№", 'Предмет', 'Тип Зан.', 'Преподаватель', 'Аудитория']  
 for i in range(5):  
 self.table1.heading(f'col{i+1}', text=col\_name[i])  
 self.table1.column(f'col{i+1}', width=len(col\_name[i]))  
   
 # Заполняем первую таблицу данными  
 table2data = self.fill\_table(self.table1, table2data, table1data)  
 self.table1.pack(fill='both', expand=True)  
 # Заголовок для первой таблицы  
 if self.label1 == None:  
 self.label1 = tk.Label(self.frame1, text=table1data.group, font=('Arial', 14, 'bold'))  
 else:  
 self.label1.destroy()  
 self.label1 = tk.Label(self.frame1, text=table1data.group, font=('Arial', 14, 'bold'))  
 self.label1.pack(pady=(0, 10))  
  
  
 # Разделитель между таблицами  
 if self.separator == None:  
 self.separator = ttk.Separator(self.new\_window, orient='horizontal')  
 self.separator.pack(fill='x', padx=20, pady=20)  
   
  
  
 # Создаем фрейм для второй таблицы  
 if self.frame2 == None:  
 self.frame2 = tk.Frame(self.new\_window)  
 self.frame2.pack(pady=20, padx=10, fill='both', expand=True)  
   
 # Вторая таблица  
 if self.table2 == None:  
 self.table2 = ttk.Treeview(self.frame2, columns=('col1', 'col2', 'col3', 'col4', 'col5'), show='headings', height=12)  
   
 # Настраиваем заголовки столбцов  
 for i in range(5):  
 self.table2.heading(f'col{i+1}', text=f'Колонка {i+1}')  
 self.table2.column(f'col{i+1}', width=120)  
   
 # Заполняем вторую таблицу данными  
 table1data = self.fill\_table(self.table2, table1data, table2data)  
 self.table2.pack(fill='both', expand=True)  
  
 # Заголовок для первой таблицы  
 # Заголовок для второй таблицы  
 if self.label2 == None:  
 self.label2 = tk.Label(self.frame2, text=table2data.group, font=('Arial', 14, 'bold'))  
 else:  
 self.label2.destroy()  
 self.label2 = tk.Label(self.frame2, text=table2data.group, font=('Arial', 14, 'bold'))  
 self.label2.pack(pady=(0, 10))  
  
 def fill\_table(self, table, table\_data\_check, table\_data: schedule):  
 """Заполняет таблицу данными с белым фоном"""  
 # Очищаем таблицу  
 for item in table.get\_children():  
 table.delete(item)  
   
 # Настраиваем белый цвет по умолчанию  
 # table.tag\_configure('white', background='white')  
   
 # Заполняем таблицу  
 print("--------------- наполенние ---------------------")  
 print(table\_data.days\_even.keys())  
 print(table\_data\_check.days\_even.keys())  
 count = 0  
 if self.parity :  
 week = table\_data.days\_even  
 week\_check = table\_data\_check.days\_even  
 else:  
 week = table\_data.days\_uneven  
 week\_check = table\_data\_check.days\_uneven  
 for d in week.keys():  
 print(week[d].pairs.keys())  
 for p in week[d].pairs.keys():  
 week[d].pairs[p].check(week\_check[d].pairs[p])  
 if p == "1":  
 week\_day = d  
 count += 1  
 else:  
 week\_day = ''  
 values = [week\_day, p, week[d].pairs[p].lesson  
 , week[d].pairs[p].type  
 , week[d].pairs[p].teacher  
 , week[d].pairs[p].auditory]  
 print(values)  
 colors = {"Identical" : ['green3', 'green2'],   
 "Similar" : ['LightGoldenrod2', 'khaki1'],  
 "Different" : ['coral1', 'salmon']}   
 if discrepancy\_to\_show[week[d].pairs[p].discrepancy]:  
 this\_tag = week[d].pairs[p].discrepancy  
 else:  
 this\_tag = 'White'  
 table.insert('', 'end', values=values, tags= this\_tag + str(count%2))  
  
 table.tag\_configure('Identical0', background='green3')  
 table.tag\_configure('Identical1', background='green2')  
 table.tag\_configure('Similar0', background='LightGoldenrod2')  
 table.tag\_configure('Similar1', background='khaki1')  
 table.tag\_configure('Different0', background='coral1')  
 table.tag\_configure('Different1', background='salmon')  
 table.tag\_configure('White0', background='White')  
 table.tag\_configure('White1', background='light gray')  
 return table\_data\_check  
 '''  
 PaleGreen2 / green2  
 tan1 / khaki1  
 coral1 / salmon  
  
 '''  
 # def change\_colors\_table1(self):  
  
 # def change\_colors\_table2(self):  
  
   
 # def change\_colors\_both(self):  
   
 # def change\_colors(self, table, color\_palette):  
  
 # def reset\_colors(self):  
  
 def create\_buttons(self, listtable1: list[schedule], listtable2: list[schedule]):  
 """Создает кнопки для управления цветами"""  
 button\_frame = tk.Frame(self.new\_window)  
 button\_frame.pack(pady=10)  
 self.groups = []  
 self.groups1 = []  
 for i in listtable1:  
 self.groups.append(i.group)   
 for i in listtable2:  
 self.groups1.append(i.group)  
  
 self.file\_combobox\_group1 = ttk.Combobox(self.new\_window, values=self.groups, state="readonly", font=(20), cursor="hand2")  
 self.file\_combobox\_group1.place(relx=1, rely=0.4, height=30, width=100, anchor="ne")  
 self.file\_combobox\_group1.bind("<<ComboboxSelected>>", self.on\_group\_selected)  
  
  
 self.file\_combobox\_group2 = ttk.Combobox(self.new\_window, values=self.groups1, state="readonly", font=(20), cursor="hand2")  
 self.file\_combobox\_group2.place(relx=1, rely=0.9, height=30, width=100, anchor="ne")  
 self.file\_combobox\_group2.bind("<<ComboboxSelected>>", self.on\_group\_selected1)  
  
 close\_btn = tk.Button(button\_frame, text="Закрыть",   
 command=self.new\_window.destroy, font=('Arial', 10), bg='#333', fg='white')  
 close\_btn.pack(side='left', padx=5)  
  
 remake\_btn = tk.Button(button\_frame, text="пересоздать",   
 command=self.create\_tables, font=('Arial', 10), bg='#333', fg='white')  
 remake\_btn.pack(side='left', padx=5)  
  
 self.click\_parity = tk.Button(button\_frame, text="ченетная",   
 command=self.on\_button\_click\_parity, font=('Arial', 10), bg='#333', fg='white')  
 self.click\_parity.pack(side='left', padx=5)  
 # self.selected\_group = ''  
 # self.selected\_group1 = ''  
  
 def on\_group\_selected(self, event):  
 # Обработчик выбора файла  
 self.selected\_group = self.file\_combobox\_group1.get()  
 if self.selected\_group:  
 print(f"выбран файл: {self.selected\_group}")  
 self.groupid1 = self.groups.index(self.selected\_group)  
 self.create\_tables()  
  
  
 def on\_group\_selected1(self, event):  
 # Обработчик выбора файла  
 self.selected\_group1 = self.file\_combobox\_group2.get()  
 if self.selected\_group1:  
 print(f"выбран файл: {self.selected\_group1}")  
 self.groupid2 = self.groups1.index(self.selected\_group1)  
 self.create\_tables()  
  
 def on\_button\_click\_parity(self): # кнопка выбора 2  
 print("переключение")  
 if self.parity == 0:  
 self.click\_parity.config(text="четная")  
 self.parity = 1  
 else:  
 self.click\_parity.config(text="ченетная")  
 self.parity = 0  
  
data\_directory = "D:\\projects\\VisualStudioCode\\Laba\_2\_3\_5\_Graphics\\data"  
selected\_file = "D:\\projects\\VisualStudioCode\\Laba\_2\_3\_5\_Graphics\\data\\iait\_17-18-1.06.02.xls"  
selected\_file1 = "D:\\projects\\VisualStudioCode\\Laba\_2\_3\_5\_Graphics\\data\\iait\_17-18-1.09.22.xls"  
def create\_window():  
 # Создаем новое окно  
 new\_window = tk.Toplevel()  
 TableWindow(new\_window, parsing(selected\_file), parsing(selected\_file1))  
  
'''//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////'''  
  
def get\_files\_in\_directory(directory="."):  
 # Возвращает список файлов в указанной директории  
 if not os.path.exists(directory):  
 return []  
   
 try:  
 items = os.listdir(directory)  
 return [item for item in items if os.path.isfile(os.path.join(directory, item))]  
 except PermissionError:  
 return []  
  
def on\_file\_selected(event):  
 # Обработчик выбора файла  
 global selected\_file  
 selected\_file = data\_directory + "\\" + file\_combobox.get()  
 if selected\_file:  
 print(f"выбран файл: {selected\_file}")  
  
def on\_file\_selected1(event):  
 # Обработчик выбора файла  
 global selected\_file1  
 selected\_file1 = data\_directory + "\\" + file\_combobox1.get()  
 if selected\_file1:  
 print(f"выбран файл: {selected\_file1}")  
  
x1 = 0  
x2 = 0  
x3 = 0  
discrepancy\_to\_show = {'Identical' : 0, 'Similar' : 0, 'Different' : 0}  
def on\_button\_click2(): # кнопка выбора 2  
 print("Сравнить сходства")  
 global x1  
 if x1 == 0:  
 btn2.config(background="lightgreen")  
 x1 =+ 1  
 discrepancy\_to\_show['Identical'] = 1  
 else:  
 btn2.config(background="lightsteelblue")  
 x1 = 0  
 discrepancy\_to\_show['Identical'] = 0  
   
def on\_button\_click3(): # кнопка выбора 3  
 print("Сравнить различия")  
 global x2  
 if x2 == 0:  
 btn3.config(background="lightgreen")  
 x2 =+ 1  
 discrepancy\_to\_show['Similar'] = 1  
 else:  
 btn3.config(background="lightsteelblue")  
 x2 = 0  
 discrepancy\_to\_show['Similar'] = 0  
   
def on\_button\_click4(): # кнопка выбора 4  
 print("Сравнить несоотствие")  
 global x3  
 if x3 == 0:  
 btn4.config(background="lightgreen")  
 x3 =+ 1  
 discrepancy\_to\_show['Different'] = 1  
 else:  
 btn4.config(background="lightsteelblue")  
 x3 = 0  
 discrepancy\_to\_show['Different'] = 0  
  
# окно  
window = tk.Tk()  
window.title("название\_окна")  
window.geometry("602x220") # размеры окна  
# window.minsize(605, 270)  
# window.maxsize(1920, 1080)  
  
# Загрузка и установка фонового изображения  
current\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))  
file\_path = os.path.join(current\_dir, "gradient\\gradient\_1920\_1080.png") # путь к фону  
file\_path1 = os.path.join(current\_dir, "gradient\\button.png") # путь к кнопке  
print(f"путь к изображению фона {file\_path}")  
print(f"путь к изображению кнопки {file\_path1}")  
bg\_image = Image.open(file\_path)  
bg\_photo = ImageTk.PhotoImage(bg\_image)  
bg\_image1 = Image.open(file\_path1)  
bg\_photo1 = ImageTk.PhotoImage(bg\_image1)  
  
# фон  
canvas = tk.Canvas(window, width=bg\_image.width, height=bg\_image.height)  
canvas.pack(fill="both", expand=True)  
  
# Добавление фонового изображения на Canvas  
canvas.create\_image(0, 0, image=bg\_photo, anchor="nw")  
  
# Сохраняем ссылку на изображение  
window.photo = bg\_photo  
window.photo = bg\_photo1  
  
# Получаем список файлов в текущей директории  
files = get\_files\_in\_directory(data\_directory)  
  
# виджеты  
btn1 = tk.Button(canvas, image=bg\_photo1, cursor="hand2", command=create\_window, activebackground="lightgreen", background="#c1fdc0") # кнопка  
btn1.place(relx=0.7, rely=0.5, anchor="nw")  
  
tk.Label(window, text=" Выберите файл для сранения ", font=(20), background="lightsteelblue1").place(anchor="sw", relx=0, rely=0.1)  
  
file\_combobox = ttk.Combobox(window, values=files, state="readonly", font=(20), cursor="hand2")  
file\_combobox.place(relx=0, rely=0.1, height=30, width=300, anchor="nw")  
file\_combobox.bind("<<ComboboxSelected>>", on\_file\_selected)  
  
tk.Label(window, text=" Выберите файл для сранения ", font=(20), background="lightsteelblue1").place(anchor="se", relx=1, rely=0.1)  
  
file\_combobox1 = ttk.Combobox(window, values=files, state="readonly", font=(20), cursor="hand2")  
file\_combobox1.place(relx=1, rely=0.1, height=30, width=300, anchor="ne")  
file\_combobox1.bind("<<ComboboxSelected>>", on\_file\_selected1)  
  
btn2 = tk.Button(canvas, text="Указать сходства", cursor="hand2", command=on\_button\_click2, activebackground="lightgray", background="lightsteelblue", font=(20)) # кнопка  
btn2.place(relx=0.3, rely=0.5, anchor="s")  
  
btn3 = tk.Button(canvas, text="Указать различия", cursor="hand2", command=on\_button\_click3, activebackground="lightgray", background="lightsteelblue", font=(20)) # кнопка  
btn3.place(relx=0.3, rely=0.6, anchor="center")  
  
btn4 = tk.Button(canvas, text="Указать несоответствие", cursor="hand2", command=on\_button\_click4, activebackground="lightgray", background="lightsteelblue", font=(20)) # кнопка  
btn4.place(relx=0.3, rely=0.7, anchor="n")  
  
window.mainloop() # это запуск, он должен быть в конце **Из файла ...ba\_2\_3\_5\_Graphics\main.py.cpp**import sys  
import os  
import xlrd  
import copy  
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))))  
from laba\_logging import \*  
  
InitFile(\_\_file\_\_)  
  
  
class pair:  
 def \_\_init\_\_(self, lesson, type, teacher, auditory):  
 self.lesson = lesson  
 self.type = type  
 self.teacher = teacher  
 self.auditory = auditory  
 def check(self, pair):  
 if self.lesson == pair.lesson:  
 if(self.type == pair.type) and (self.teacher == pair.teacher) and (self.auditory == pair.auditory):  
 self.discrepancy = "Identical"  
 pair.discrepancy = "Identical"  
 print("сравнение - Identical")  
 else:  
 self.discrepancy = "Similar"  
 pair.discrepancy = "Similar"  
 print("сравнение - Similar")  
  
 else:  
 self.discrepancy = "Different"  
 pair.discrepancy = "Different"  
 print("сравнение - Different")  
  
 def tostr(self):  
 return f"pair {self.lesson}, {self.type}, {self.teacher}, {self.auditory}"  
  
  
class day:  
 def \_\_init\_\_(self):  
 self.pairs = {}  
 def add\_pair(self, num, lesson, type, teacher, auditory):  
 self.pairs[num] = pair(lesson, type, teacher, auditory)  
 # print(f"add\_pair {num}, {lesson}, {type}, {teacher}, {auditory}")  
 # print(f"self.pairs[num] {self.pairs[num].tostr()}")  
 # print(f"self.pairs.keys() {self.pairs.keys()}")  
 def tostr(self):  
 tmp = 'day'  
 for i in self.pairs.keys():  
 tmp += f"\n{i} - {self.pairs[i].tostr()}"  
 return tmp  
  
class schedule:  
 def \_\_init\_\_(self, group, year, semester):  
 self.group = group  
 self.year = year  
 self.semester = semester  
 self.days\_even = {}  
 self.days\_uneven = {}  
   
 def add\_day(self, week\_day, day, ev):  
 if ev:  
 self.days\_even[week\_day] = copy.deepcopy(day)   
 else:   
 self.days\_uneven[week\_day] = copy.deepcopy(day)   
 def print\_data(self):  
 print(f"check days days\_even: {self.days\_even.keys()}")  
 for i in self.days\_even.keys():  
 print(f"day {i} / {self.days\_even[i].tostr()}")  
 print(f"check days days\_uneven: {self.days\_uneven.keys()}")  
 for i in self.days\_uneven.keys():  
 print(f"day {i} / {self.days\_uneven[i].tostr()}")  
# iait\_17-18-1.09.22  
def parsing(path = 'D:\\projects\\VisualStudioCode\\Laba\_2\_3\_5\_Graphics\\data\\iait\_17-18-1.06.02.xls'):  
 # Открытие файла  
 print("parsing...")  
 workbook = xlrd.open\_workbook(path)  
 sheet = workbook.sheet\_by\_index(0)  
 group = ''  
 schedules = []  
 tmp\_schedule = schedule(sheet.cell\_value(4, 0), '0', '0')  
 last\_week\_day = ""  
 # Чтение данных  
 count = 0  
 day\_ev = day()  
 day\_nev = day()  
 for row in range(4, sheet.nrows):  
 count += 1  
 for col in range(1, sheet.ncols):  
 print(sheet.cell\_value(row, col), end="|")  
 print('', end="\n")  
 # print(f"new str {sheet.cell\_value(row, 0)}|{sheet.cell\_value(row, 1)}|{sheet.cell\_value(row, 2)}")  
  
 if(sheet.cell\_value(row, 1) != ""):  
 last\_week\_day = sheet.cell\_value(row, 1)  
  
  
 print(f"{row} of {(sheet.nrows)-1}")  
 if(sheet.cell\_value(row, 2) == "") or (row >= (sheet.nrows-1)):  
 while (6 >= count):  
 day\_ev .add\_pair(str(count), '', '', '', '')  
 day\_nev.add\_pair(str(count), '', '', '', '')  
 count += 1  
 # print(f"end of day: \n{str(day\_ev.pairs['1'].lesson)}\n{str(day\_nev.pairs['1'].lesson)}")  
 tmp\_schedule.add\_day(last\_week\_day, day\_ev, 0)  
 tmp\_schedule.add\_day(last\_week\_day, day\_nev, 1)  
 day\_ev .pairs.clear()  
 day\_nev.pairs.clear()  
 count = 0  
 else:  
 while (int(sheet.cell\_value(row, 2)) > count):  
 day\_ev .add\_pair(str(count), '', '', '', '')  
 day\_nev.add\_pair(str(count), '', '', '', '')  
 count += 1  
 day\_ev .add\_pair(str(count), sheet.cell\_value(row, 3), sheet.cell\_value(row, 5), sheet.cell\_value(row, 6), sheet.cell\_value(row, 7))  
 day\_nev.add\_pair(str(count), sheet.cell\_value(row, 10), sheet.cell\_value(row, 12), sheet.cell\_value(row, 13), sheet.cell\_value(row, 14))  
  
  
  
 if(sheet.cell\_value(row, 0) != '') or (row == sheet.nrows-1):  
 # print(f"old | new group: {group}|{sheet.cell\_value(row, 0)}")  
 if group != "":  
 print("group: " + group)  
 for day\_name in ['ПН', 'ВТ', 'СР', 'ЧТ', 'ПТ', 'СБ']:  
 if day\_name not in tmp\_schedule.days\_even.keys():  
 for i in range(1, 7):  
 day\_ev .add\_pair(str(i), '', '', '', '')  
 day\_nev.add\_pair(str(i), '', '', '', '')  
 tmp\_schedule.add\_day(day\_name, day\_ev, 0)  
 tmp\_schedule.add\_day(day\_name, day\_nev, 1)  
 day\_ev .pairs.clear()  
 day\_nev.pairs.clear()  
 schedules.append(tmp\_schedule)  
 tmp\_schedule.print\_data()  
 tmp\_schedule = schedule(group, '0', '0')  
 count = 0  
 group = sheet.cell\_value(row, 0)  
  
  
  
  
  
 # for col in range(1, sheet.ncols):  
 # print(f'[{row}][{col}] - [{sheet.cell\_value(row, col)}]')  
 return schedules  
 '''  
 # Для pandas  
 pip install pandas openpyxl xlrd  
  
 # Или только нужные компоненты  
 pip install pandas  
 pip install openpyxl # для .xlsx файлов  
 pip install xlrd # для старых .xls файлов  
 '''  
parsing()  
Снимок.PNG:  
  
  
 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')