Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ЛАБАРАТОРНОЙ РАБОТЕ №9**

**«РАБОТА С ТИПИЗИРОВАННЫМИ ФАЙЛАМИ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»**

Выполнил: студент учебной группы

ИСПк-205-52-00

Юдинцев Богдан Сергеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

В отчете должны отображаться:

1. Цель лабораторной работы
2. Формулировка задания (с вариантом)
3. Схема алгоритма с комментариями
4. Код программы
5. Результат выполнения программы
6. Вывод

Цель работы: получить представление об организации хранения и использования информации посредством типизированных файлов, закрепить навыки создания пользовательского интерфейса.

Задание:

1. Сформировать структуру записи, с которой будет осуществляться работа, и согласовать ее с преподавателем.

2. Разработать приложение, позволяющее извлекать набор записей из типизированного файла, визуализировать данный набор, а также позволяющее изменять данных и выполнять сохранение проделанных изменений.

Схема программы:

start

|

v

create\_widgets

|

v

load\_groups\_from\_database

|

v

add\_group

|

v

save\_groups\_to\_database

|

v

add\_student

|

v

save\_groups\_to\_database

|

v

on\_group\_select

|

v

endТекст 1 – Схема программа

Код программы 1:

import tkinter as tk

from tkinter import messagebox

import sqlite3

class Group:

    def \_\_init\_\_(self, teacher, name):

        self.teacher = teacher

        self.name = name

        self.students = []

class Application:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Информационное приложение")

        self.create\_widgets()

        self.groups = []

        self.load\_groups\_from\_database()

        self.selected\_group\_index = None

    def create\_widgets(self):

        self.groups\_frame = tk.Frame(self.root)

        self.groups\_frame.pack(side=tk.LEFT, padx=10)

        self.teacher\_label = tk.Label(self.groups\_frame, text="Учитель:")

        self.teacher\_label.pack()

        self.teacher\_entry = tk.Entry(self.groups\_frame)

        self.teacher\_entry.pack()

        self.group\_name\_label = tk.Label(self.groups\_frame, text="Название группы:")

        self.group\_name\_label.pack()

        self.group\_name\_entry = tk.Entry(self.groups\_frame)

        self.group\_name\_entry.pack()

        self.add\_group\_button = tk.Button(self.groups\_frame, text="Добавить группу", command=self.add\_group)

        self.add\_group\_button.pack()

        self.listbox = tk.Listbox(self.groups\_frame, width=30)

        self.listbox.pack()

        self.listbox.bind("<<ListboxSelect>>", self.on\_group\_select)

        self.students\_frame = tk.Frame(self.root)

        self.students\_frame.pack(side=tk.LEFT, padx=10)

        self.students\_label = tk.Label(self.students\_frame, text="Ученики:")

        self.students\_label.pack()

        self.students\_text = tk.Text(self.students\_frame, height=10, width=30, state='disabled')

        self.students\_text.pack()

        self.add\_student\_label = tk.Label(self.students\_frame, text="Добавить ученика:")

        self.add\_student\_label.pack()

        self.add\_student\_entry = tk.Entry(self.students\_frame)

        self.add\_student\_entry.pack()

        self.add\_student\_button = tk.Button(self.students\_frame, text="Добавить", command=self.add\_student)

        self.add\_student\_button.pack()

    def save\_groups\_to\_database(self):

        conn = sqlite3.connect("database.db")

        cursor = conn.cursor()

        cursor.execute("""

            CREATE TABLE IF NOT EXISTS groups (

                id INTEGER PRIMARY KEY,

                teacher TEXT,

                name TEXT

            )

        """)

        cursor.execute("""

            CREATE TABLE IF NOT EXISTS students (

                id INTEGER PRIMARY KEY,

                group\_id INTEGER,

                name TEXT,

                FOREIGN KEY (group\_id) REFERENCES groups (id)

            )

        """)

        cursor.execute("DELETE FROM groups")

        cursor.execute("DELETE FROM students")

        for group in self.groups:

            cursor.execute("INSERT INTO groups (teacher, name) VALUES (?, ?)", (group.teacher, group.name))

            group\_id = cursor.lastrowid

            for student in group.students:

                cursor.execute("INSERT INTO students (group\_id, name) VALUES (?, ?)", (group\_id, student))

        conn.commit()

        conn.close()

    def load\_groups\_from\_database(self):

        conn = sqlite3.connect("database.db")

        cursor = conn.cursor()

        cursor.execute("SELECT name FROM sqlite\_master WHERE type='table' AND name='groups'")

        table\_exists = cursor.fetchone()

        if not table\_exists:

            self.save\_groups\_to\_database()

        else:

            cursor.execute("SELECT id, teacher, name FROM groups")

            rows = cursor.fetchall()

            for row in rows:

                group\_id = row[0]

                teacher = row[1]

                name = row[2]

                group = Group(teacher, name)

                cursor.execute("SELECT name FROM students WHERE group\_id=?", (group\_id,))

                students = cursor.fetchall()

                for student\_row in students:

                    student = student\_row[0]

                    group.students.append(student)

                self.groups.append(group)

                self.listbox.insert(tk.END, f"{name} - {teacher}")

        conn.close()

    def add\_group(self):

        teacher = self.teacher\_entry.get()

        group\_name = self.group\_name\_entry.get()

        if not teacher or not group\_name:

            messagebox.showerror("Ошибка", "Введите имя учителя и название группы")

            return

        for group in self.groups:

            if group.teacher == teacher and group.name == group\_name:

                messagebox.showerror("Ошибка", "Группа с таким учителем и названием уже существует")

                return

        group = Group(teacher, group\_name)

        self.groups.append(group)

        self.listbox.insert(tk.END, f"{group\_name} - {teacher}")

        self.teacher\_entry.delete(0, tk.END)

        self.group\_name\_entry.delete(0, tk.END)

        self.save\_groups\_to\_database()

    def add\_student(self):

        if self.selected\_group\_index is None:

            messagebox.showerror("Ошибка", "Выберите группу")

            return

        group = self.groups[self.selected\_group\_index]

        student = self.add\_student\_entry.get()

        if not student:

            messagebox.showerror("Ошибка", "Введите имя ученика")

            return

        student\_number = len(group.students) + 1

        group.students.append(student)

        self.students\_text.config(state='normal')

        self.students\_text.insert(tk.END, f"{student\_number}. {student}\n")

        self.students\_text.config(state='disabled')

        self.add\_student\_entry.delete(0, tk.END)

        self.save\_groups\_to\_database()

    def on\_group\_select(self, event):

        selected\_index = self.listbox.curselection()

        if not selected\_index:

            self.selected\_group\_index = None

            return

        selected\_index = selected\_index[0]

        self.students\_text.config(state='normal')

        self.students\_text.delete("1.0", tk.END)

        group = self.groups[selected\_index]

        for i, student in enumerate(group.students, start=1):

            self.students\_text.insert(tk.END, f"{i}. {student}\n")

        self.students\_text.config(state='disabled')

        self.add\_student\_entry.delete(0, tk.END)

        self.selected\_group\_index = selected\_index

    def run(self):

        self.root.mainloop()

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

    root = tk.Tk()

    app = Application(root)

    app.run()

Вывод:

Была сделана лабораторная работа №8. Данная программа реализует простой таймер с использованием библиотеки Tkinter в языке программирования Python. Она предоставляет пользователю возможность выбирать время в часах, минутах и секундах, запускать, останавливать и сбрасывать таймер.