Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ЛАБАРАТОРНОЙ РАБОТЕ №8**

**«ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ФУНКЦИИ»**

**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: получение базовых навыков реализации приложений с графическим интерфейсом пользователя на основе событийно-ориентированной парадигмы.

Задание:

1. Реализовать приложение "Таймер".

2. Предусмотреть возможность подачи звукового сигнала по истечении заданного промежутка

времени.

3. Предусмотреть возможность досрочной остановки таймера.

Схема программы:

Начинать

├── Инициализировать Tkinter и импортировать необходимые модули

│ ├── Создать класс Timer

│ │ ├── Инициализировать класс Timer с главным окном и установить начальные значения

│ │ ├── Создание объектов Tkinter StringVar для часов, минут и секунд

│ │ ├── Создание меток для отображения времени

│ │ ├── Создание кнопок для запуска, остановки и сброса действий

│ │ ├── Создание выпадающих списков для выбора часов, минут и секунд

│ │ ├── Определение функции start\_timer

│ │ │ ├── Проверьте, не запущен ли таймер

│ │ │ ├── Получить выбранные значения из выпадающих списков, если таймер не установлен

│ │ │ ├── Вызвать функцию update\_time

│ │ ├── Определение функции stop\_timer

│ │ │ ├── Проверить, работает ли таймер

│ │ │ ├── Отменить таймер, если он существует

│ │ ├── Определение функции reset\_timer

│ │ │ ├── Вызов функции stop\_timer

│ │ │ ├── Сбросить время на [0, 0, 0]

│ │ │ ├── Вызвать функцию update\_display

│ │ ├── Определение функции play\_sound

│ │ │ ├── Воспроизведение звука с помощью модуля winsound

│ │ ├── Определение функции update\_time

│ │ │ ├── Проверить, равно ли время [0, 0, 0]

│ │ │ │ ├── Вызвать функцию play\_sound

│ │ │ │ ├── Вызов функции stop\_timer

│ │ │ ├── Остальное

│ │ │ │ ├── Проверить, работает ли таймер

│ │ │ │ ├── Обновление значений времени на основе логики обратного отсчета

│ │ │ ├── Вызвать функцию update\_display

│ │ │ ├── Запланировать вызов функции update\_time через 1000 миллисекунд (1 секунду)

│ │ ├── Определение функции update\_display

│ │ │ ├── Обновить объекты StringVar текущими значениями времени

│ ├── Создать корневое окно Tkinter

│ ├── Создать экземпляр класса Timer с корневым окном

├── Запустить основной цикл событий Tkinter

Конец

Текст 1 – Схема программа

Код программы 1:

import tkinter as tk

from tkinter import ttk

import winsound

class Timer:

    def \_\_init\_\_(self, master):

        self.master = master

        self.master.title("Таймер")

        self.time = [0, 0, 0]

        self.is\_running = False

        self.timer\_id = None

        self.hours\_var = tk.StringVar(value='00')

        self.minutes\_var = tk.StringVar(value='00')

        self.seconds\_var = tk.StringVar(value='00')

        self.create\_widgets()

    def create\_widgets(self):

        self.time\_label = tk.Label(self.master, textvariable=self.hours\_var)

        self.time\_label.grid(row=0, column=0, padx=10, pady=10)

        self.time\_label = tk.Label(self.master, text=":")

        self.time\_label.grid(row=0, column=1)

        self.time\_label = tk.Label(self.master, textvariable=self.minutes\_var)

        self.time\_label.grid(row=0, column=2, padx=10, pady=10)

        self.time\_label = tk.Label(self.master, text=":")

        self.time\_label.grid(row=0, column=3)

        self.time\_label = tk.Label(self.master, textvariable=self.seconds\_var)

        self.time\_label.grid(row=0, column=4, padx=10, pady=10)

        self.start\_button = tk.Button(self.master, text="Старт", command=self.start\_timer)

        self.start\_button.grid(row=1, column=0, padx=10, pady=10)

        self.stop\_button = tk.Button(self.master, text="Стоп", command=self.stop\_timer)

        self.stop\_button.grid(row=1, column=1, padx=10, pady=10)

        self.reset\_button = tk.Button(self.master, text="Сбросить", command=self.reset\_timer)

        self.reset\_button.grid(row=1, column=2, padx=10, pady=10)

        self.hours\_combobox = ttk.Combobox(self.master, values=list(range(24)), width=2, state="readonly")

        self.hours\_combobox.current(0)

        self.hours\_combobox.grid(row=2, column=0, padx=10, pady=10)

        self.minutes\_combobox = ttk.Combobox(self.master, values=list(range(60)), width=2, state="readonly")

        self.minutes\_combobox.current(0)

        self.minutes\_combobox.grid(row=2, column=2, padx=10, pady=10)

        self.seconds\_combobox = ttk.Combobox(self.master, values=list(range(60)), width=2, state="readonly")

        self.seconds\_combobox.current(0)

        self.seconds\_combobox.grid(row=2, column=4, padx=10, pady=10)

    def start\_timer(self):

        if not self.is\_running:

            self.is\_running = True

            if self.time == [0, 0, 0]:

                self.time = [

                    int(self.hours\_combobox.get()),

                    int(self.minutes\_combobox.get()),

                    int(self.seconds\_combobox.get())

                ]

            self.update\_time()

    def stop\_timer(self):

        if self.is\_running:

            self.is\_running = False

            if self.timer\_id:

                self.master.after\_cancel(self.timer\_id)

                self.timer\_id = None

    def reset\_timer(self):

        self.stop\_timer()

        self.time = [0, 0, 0]

        self.update\_display()

    def play\_sound(self):

        winsound.Beep(1000, 1000)

    def update\_time(self):

        if self.time == [0, 0, 0]:

            self.play\_sound()

            self.stop\_timer()

        else:

            if self.is\_running:

                if self.time[2] > 0:

                    self.time[2] -= 1

                elif self.time[1] > 0:

                    self.time[2] = 59

                    self.time[1] -= 1

                elif self.time[0] > 0:

                    self.time[2] = 59

                    self.time[1] = 59

                    self.time[0] -= 1

            self.update\_display()

            self.timer\_id = self.master.after(1000, self.update\_time)

    def update\_display(self):

        self.hours\_var.set(str(self.time[0]).zfill(2))

        self.minutes\_var.set(str(self.time[1]).zfill(2))

        self.seconds\_var.set(str(self.time[2]).zfill(2))

root = tk.Tk()

timer = Timer(root)

root.mainloop()

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.