Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**Создание приложения с графическим интерфейсом пользователя**

Лабораторная работа №7

по курсу «Разработка программных систем»

Выполнил студент группы ИВТб-31 /Категов А. Д./ Проверил преподаватель /Чистяков Г.А./

Киров 2024

1. Цель

Целью работы является знакомство с библиотекой Tkinter, предназначенной для создания графического пользовательского интерфейса, а также со стандартной библиотекой языка Python.

1. Задание

Разработать с использованием элементов стандартной библиотеки приложение с графическим интерфейсом. Для выполнения лабораторной работы необходимо решить следующие задачи:

* Согласовать тематику разработки с преподавателем
* Разработать структуру приложения
* Разработать графический интерфейс пользователя
* Реализовать приложение
* Продемонстрировать работу приложения.

Приложение реализует функции для работы с простыми числами:

* Проверку числа на простоту
* Разложение числа на простые множители
* Получение случайного простого числа
* Генерацию 100 простых чисел и запись их в файл

1. Листинг программы

Листинг программной реализации приведен в приложении А.

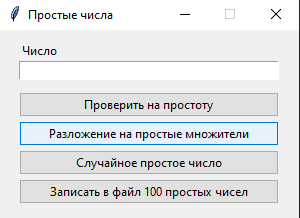


Рисунок 1 – Пользовательский интерфейс

**Вывод**: в ходе выполнения лабораторной работы был изучен основной функционал библиотеки для создания графического пользовательского интерфейса Tkinter. В процессе работы было написано приложение с графическим пользовательским интерфейсом, выполняющее функции для работы с простыми числами. На основании всего сказанного выше можно сделать вывод о том, что цель достигнута, необходимые знания получены, задание выполнено верно, что доказывается полученными результатами.

Приложение А

(обязательное)

Листинг программы

**model\_prime.py**

import random

MAX\_PRIME = 999999937

def is\_prime(n):                 # optimized trial division with 6k optimization

    def check\_divisors(d):       # check the divisors from 5 to sqrt(n) in increments of 6

        if d \* d > n:

            return True

        if n % d == 0 or n % (d + 2) == 0:

            return False

        return check\_divisors(d + 6)

    if n <= 1:                   # numbers less than or equal to 1 are not prime

        return False

    if n <= 3:                   # 2 and 3 are prime

        return True

    if n % 2 == 0 or n % 3 == 0: # exclude even numbers and numbers divisible by 3

        return False

    return check\_divisors(5)

prime\_factorization = lambda num, k = 2: [num] if is\_prime(num) else [k] + prime\_factorization(num // k, k) if num % k == 0 else prime\_factorization(num, k + 1)

get\_next = lambda i: get\_next(i + 1) if not is\_prime(i + 1) else i + 1 # find the next prime number

get\_random\_prime = lambda: get\_next(random.randint(2, MAX\_PRIME))

def generate\_primes(size): # generating an array of prime numbers

    def get\_unique\_primes(accumulated\_primes):

        if len(accumulated\_primes) >= size:

            return accumulated\_primes

        prime = get\_random\_prime()

        accumulated\_primes.append(prime)

        return get\_unique\_primes(accumulated\_primes)

    if size <= 0:

        return []

    return get\_unique\_primes([])

**main.py**

from tkinter import \*

from view import View

from controller import Controller

import model\_prime as model\_prime

class App(Tk):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title('Простые числа')

        self.geometry('300x190')

        self.resizable(False, False)

        view = View(self)

        self.controller = Controller(model\_prime, view)

        view.set\_controller(self.controller)

        view.create\_controls()

        self.grid\_columnconfigure(0, weight=1)

        self.protocol("WM\_DELETE\_WINDOW", self.on\_closing)

    def on\_closing(self):

        self.destroy()

if \_\_name\_\_ == '\_\_main\_\_':

    app = App()

    app.mainloop()

**controller.py**

class Controller:

    def \_\_init\_\_(self, model, views): # constructor

        self.prime = model

        self.views = views

    def int\_conversion(self, value):

        """conversion to int"""

        try:

            v = int(value)

            return v

        except ValueError:

            return None

    def is\_prime(self, value):

        """checking for a prime"""

        v = self.int\_conversion(value)

        if (v is None) or (v < 0):

            self.views.show\_result(1, "Введите неотрицательное целое число")

            return

        if v == 1 or v == 0:

            self.views.show\_result(0, "Число не является ни простым ни составным")

        else:

            self.views.show\_result(0, "Число простое" if self.prime.is\_prime(v) else "Число составное")

    def prime\_factorization(self, value):

        v = self.int\_conversion(value)

        if (v is None) or (v < 0):

            self.views.show\_result(1, "Введите неотрицательное целое число")

            return

        if v == 1 or v == 0:

            self.views.show\_result(0, "Число не является ни простым ни составным")

        else:

            r = self.prime.prime\_factorization(v)

            self.views.show\_result(0, f"Простые множители: {\*r,}" if len(r) > 1 else "Это простое число")

    def get\_random\_prime(self):

        self.views.set\_input\_value(self.prime.get\_random\_prime())

    def file\_name\_valid(self, value):

        """determines whether the file name is valid"""

        try:

            invalid\_chars = ['/', '\\', '|', ':', '\*', '?', '"', '<', '>']

            for char in invalid\_chars:

                if char in value:

                    return None

            if value == '':

                return None

            return value

        except ValueError:

            return None

    def generate\_primes(self, filename):

        """save primes to file"""

        file = self.file\_name\_valid(filename)

        if (file is None):

            self.views.show\_result(1, "Недопустимое название файла")

            return

        primes = self.prime.generate\_primes(100)

        with open(file + '.txt', 'w') as file:

            for prime in primes:

                file.write(f"{prime}\n")

**view.py**

from tkinter import \*

import tkinter.messagebox as messagebox

import tkinter.ttk as ttk

class View(ttk.Frame):

    def \_\_init\_\_(self, parent): #constructor

        super().\_\_init\_\_(parent)

        self.controller = None

        self.number\_input = None

        self.grid\_columnconfigure(0, weight=1)

    def set\_controller(self, controller):

        self.controller = controller

    def create\_controls(self):

        label = Label(self, text="Число")

        label.grid(row=0, column=0, padx=10, sticky="w")

        self.number\_input = Entry(self)

        self.number\_input.grid(row=1, column=0, padx=10, pady=(0, 10), sticky="we")

        check\_button = ttk.Button(self, text='Проверить на простоту', command=self.check)

        check\_button.grid(row=2, column=0, padx=10, pady=2, sticky="we")

        factorize\_button = ttk.Button(self, text='Разложение на простые множители', command=self.factorize)

        factorize\_button.grid(row=3, column=0, padx=10, pady=2, sticky="we")

        random\_prime\_button = ttk.Button(self, text='Случайное простое число', command=self.random\_prime)

        random\_prime\_button.grid(row=4, column=0, padx=10, pady=2, sticky="we")

        next\_prime\_button = ttk.Button(self, text='Записать в файл 100 простых чисел', command=self.generate\_primes)

        next\_prime\_button.grid(row=5, column=0, padx=10, pady=2, sticky="we")

        self.grid(row=0, column=0, pady=10, padx=10,  sticky="we")

    def check(self):

        if self.controller:

            if self.number\_input.get():

                self.controller.is\_prime(self.number\_input.get())

    def factorize(self):

        if self.controller:

            if self.number\_input.get():

                self.controller.prime\_factorization(self.number\_input.get())

    def random\_prime(self):

        if self.controller:

            self.controller.get\_random\_prime()

    def generate\_primes(self):

        if self.controller:

            if self.number\_input.get():

                self.controller.generate\_primes(self.number\_input.get())

    def show\_result(self, mtype, msg):

        if mtype == 0:

            messagebox.showinfo("Результат", msg)

        elif mtype == 1:

            messagebox.showerror("Ошибка", msg)

    def set\_input\_value(self, v):

        self.number\_input.delete(0, END)

        self.number\_input.insert(0, v)