

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.03 | Прикладная информатика | | |
| Направленность (профиль) | |  | Прикладная информатика в химии | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Разработка программных систем | | |
| Курс | II | | | Группа | 425 |

Отчёт по лабораторной работе № 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнители: |  |  |  |  |
| обучающиеся группы 425 |  |  |  | Смирнова С. А. |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Дамрин А. О.  Соболевский В. А. |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |

СОДЕРЖАНИЕ

[1 Цель работы и задание 3](#_Toc153291233)

[Цель работы 3](#_Toc153291234)

[Задание 3](#_Toc153291235)

[2 Описание хода выполнения 4](#_Toc153291236)

[3 Результаты тестирования 4](#_Toc153291238)

[4 Описание графического интерфейса 5](#_Toc153291239)

[Вывод 8](#_Toc153291240)

# Цель работы и задание

## Цель работы

Разработка для программы сортировки графического пользовательского интерфейса и интеграция базы данных.

|  |  |  |
| --- | --- | --- |
| 7 | Сортировка Шелла / Shell sort | Смирнова Светлана Алексеевна |

## Задание

1. Для проекта, созданного в рамках 2 лабораторной, в выбранной системе контроля версий создать новую ветку (branch). Все дальнейшие модификации должны вноситься исключительно в созданную ветку, исходная ветка со 2 лабораторной не должна быть изменена. Каждый участник учебной группы в рамках работы над лабораторной должен внести хотя бы одно изменение в проект.
2. Реализовать 5 программных тестов работы с базой данных:

1) тест добавления 100 массивов в базу данных программы (размер и наполнение массивов генерируется случайным образом), который в качестве ответа выводит флаг успешного / не успешного выполнения и время работы;

2) тест добавления 1000 массивов в базу данных программы (размер и наполнение массивов генерируется случайным образом), который в качестве ответа выводит флаг успешного / не успешного выполнения и время работы;

3) тест добавления 10000 массивов в базу данных программы (размер и наполнение массивов генерируется случайным образом), который в качестве ответа выводит флаг успешного / не успешного выполнения и время работы;

4) тест выгрузки и сортировки 100 случайных массивов из базы данных, который в качестве ответа выводит флаг успешного / не успешного выполнения, общее время работы теста и среднее время работы с 1 массивом (данный тест должен быть запущен 3 раза для базы на 100, 1000 и 10000 записей);

5) тест очистки базы данных, который в качестве ответа выводит флаг успешного / не успешного выполнения и общее время работы теста (данный тест должен быть запущен 3 раза для базы на 100, 1000 и 10000 записей). Тесты должны обращаться к программным функциям создаваемой программы, но не должны быть частью самой программы.

1. Развернуть базу данных (любую на выбор, в том числе можно использовать в качестве базы данных текстовый файл, SQL или NoSQL СУБД). Создать структуру базы данных, реализующую хранение массивов (как отсортированных, так и не отсортированных). Протестировать работу базы с помощью тестов, написанных в предыдущем пункте. Результаты тестирования записать в отчёт.
2. Реализовать графический пользовательский интерфейс (стек технологий любой на выбор). Интерфейс должен иметь функции:

1) ввода/редактирования массива для сортировки с клавиатуры;

2) сохранение выведенного в интерфейсе массива в базу данных;

3) вывод списка всех массивов, хранящихся в базе (без возможности редактирования и сортировки);

4) выгрузка массива из базы для редактирования и сортировки;

5) сортировка массива по алгоритму, реализованному во 2 лабораторной.

Получившийся интерфейс должен хотя бы отчасти соответствовать следующий эвристикам ui / ux :

1. видимость статуса системы;
2. соответствие между системой и реальным миром;
3. управляемость;
4. согласованность;
5. гибкость;
6. минимализм в дизайне;
7. помощь пользователю в выявлении ошибок;
8. наличие справки.

В отчёте должны быть даны пояснения, какие элементы графического интерфейса какой эвристике соответствуют.

1. Написать отчёт, в котором представить результат работы программы и все необходимые пояснения, указанные в предыдущих пунктах работы.

* Текст программы должен быть хорошо прокомментирован. По всем не прокомментированным участкам программы будут заданы вопросы.

# Описание хода выполнения

Результат выполнения задания выложен по адресу (ветка ):

<https://github.com/SmirnovaProgramming/main/tree/RPS_3>

Для разработки графического интерфейса была использована стандартная графическая библиотека Python 3 – Tkinter. Для создания базы была выбрана SQLite, управляемая из стандартного модуля Python 3 – sqlite3.

Tkinter не имеет особых преимуществ над другими библиотеками, такими как PyQt5, но обладает достаточным функционалом для быстрого прототипирования интерфейса несложных приложений, как в данной контрольной работе.

SQLite был выбран в связи с простотой создания базы данных и её переноса данных между несколькими устройствами.

# Результаты тестирования

База данных имеет структуру, содержащую одну таблицу с двумя полями: sequence (содержит массив), type (содержит тип последовательности, raw – обычная, sorted – отсортированная).

Результат проведения тестов работы с базой данных представлен ниже. Время работы указано в секундах. Последовательности составлялись из чисел длиной до 15 знаков в количестве до 100 штук.

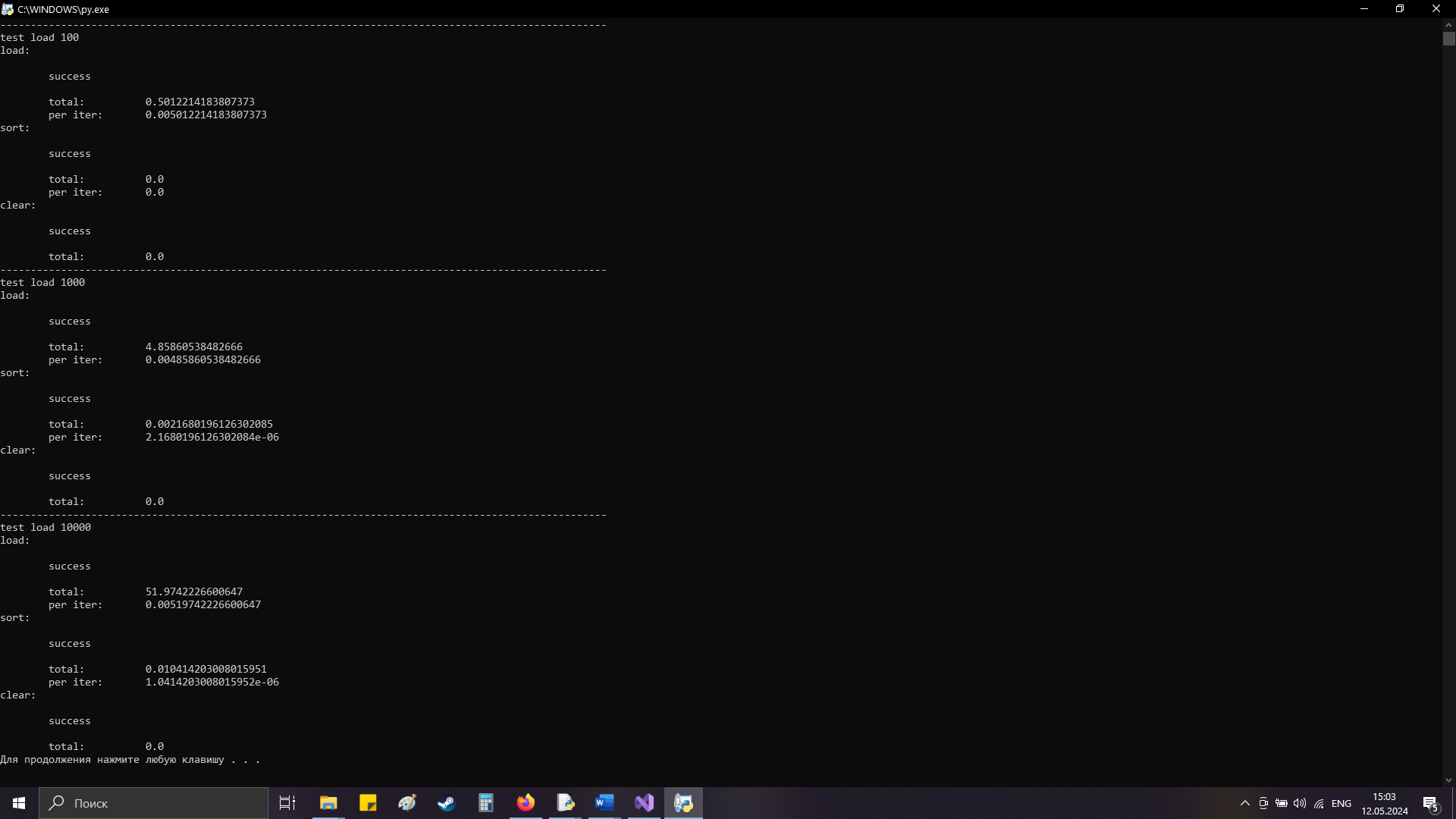


Рисунок 1 – Результаты тестирования

# Описание графического интерфейса

Основные окна представлены на рисунках 2-4.

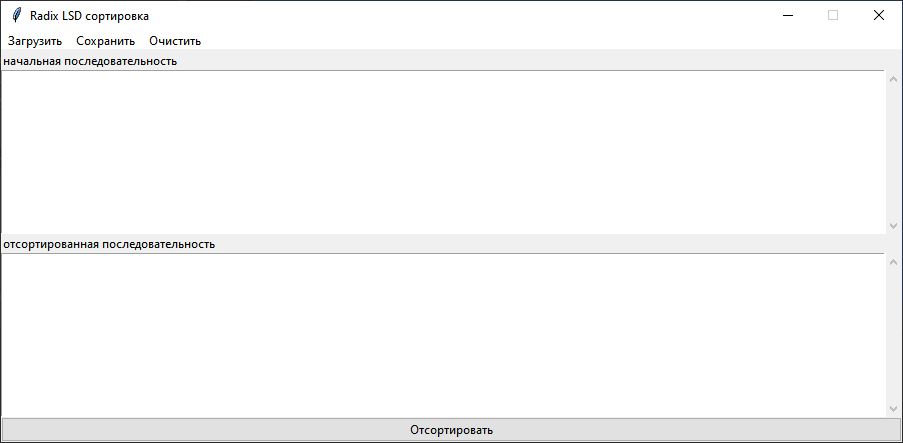
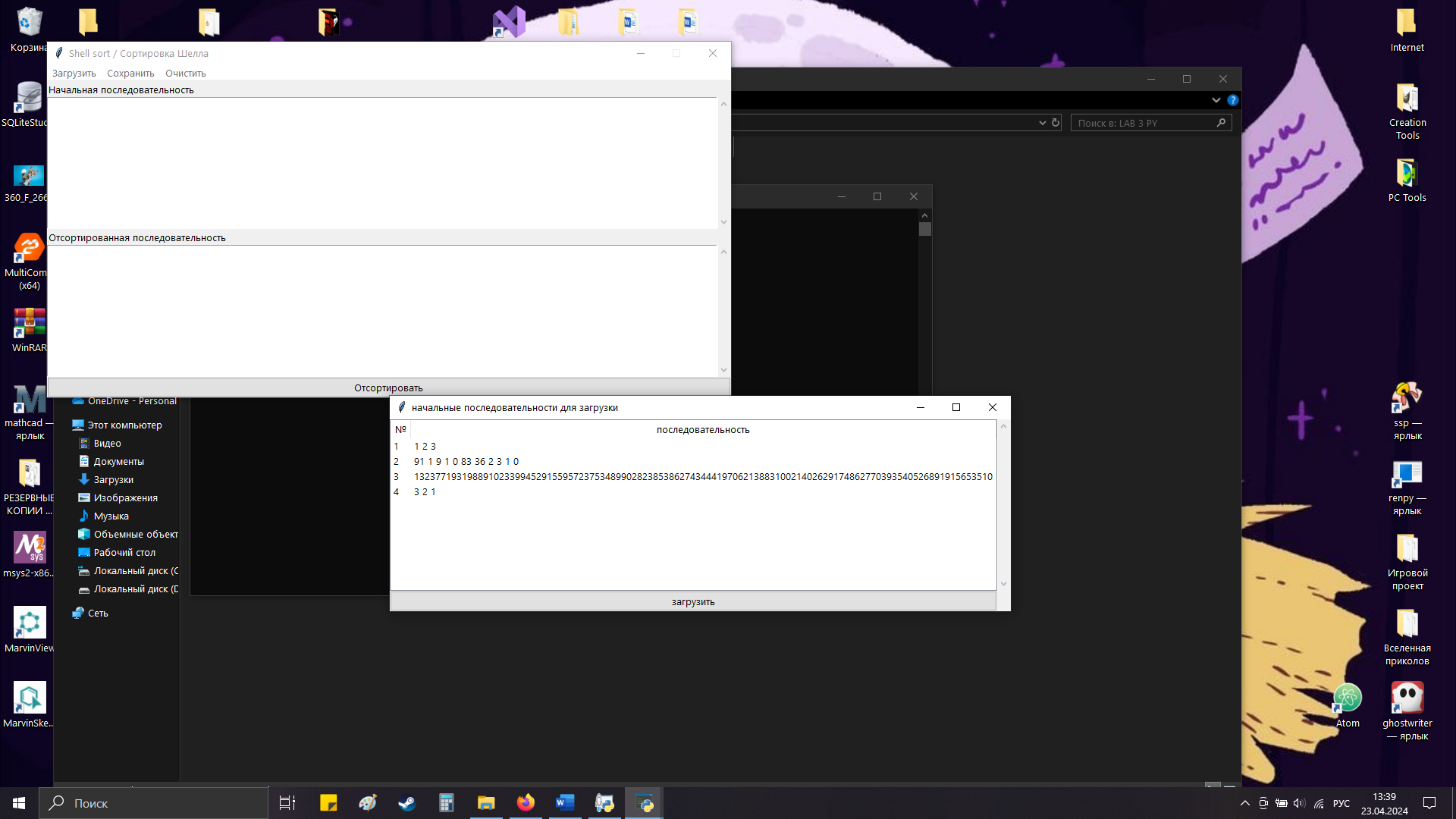


Рисунок 2 – основное окно приложения



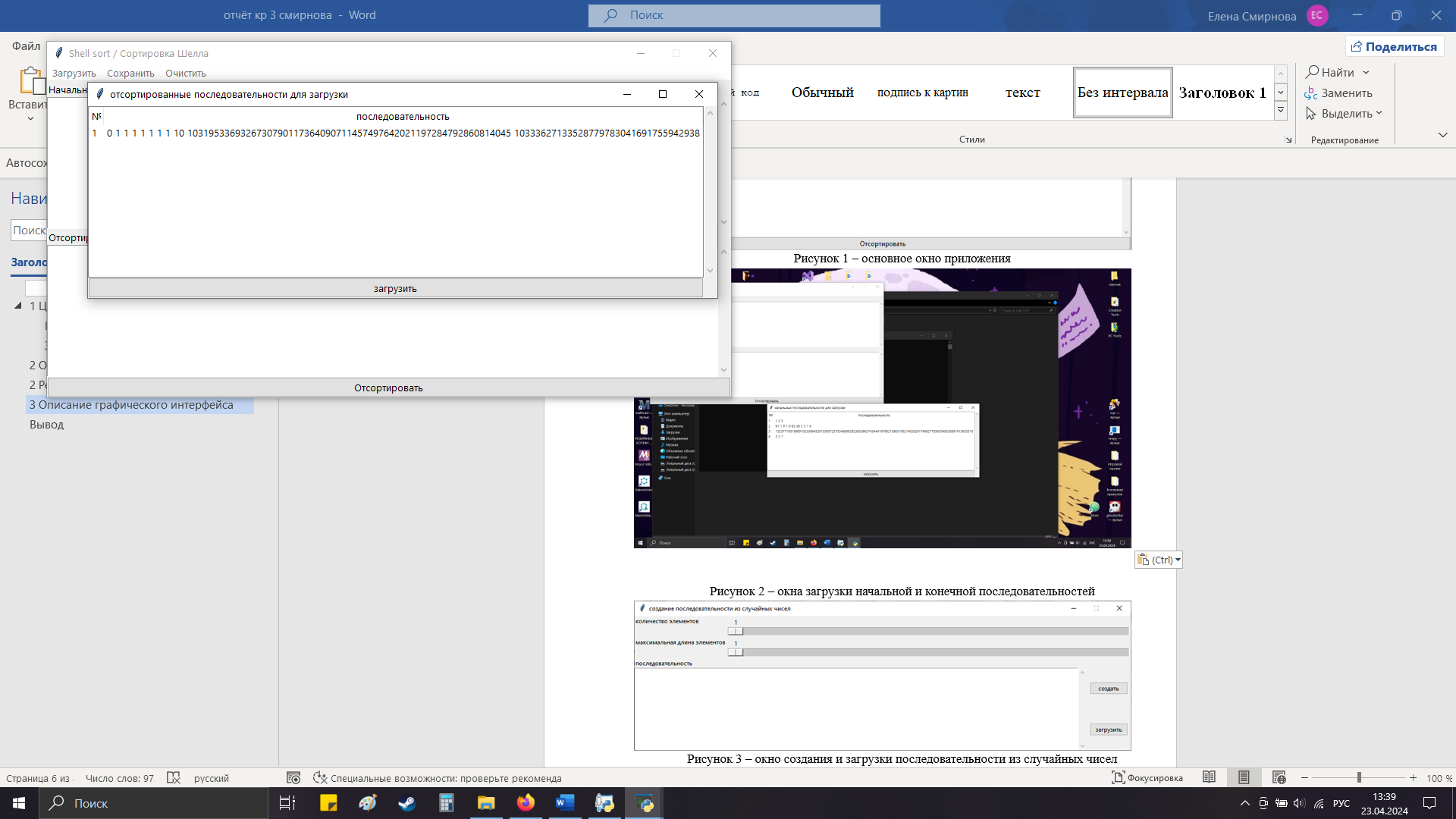


Рисунок 3 – окна загрузки начальной и конечной последовательностей

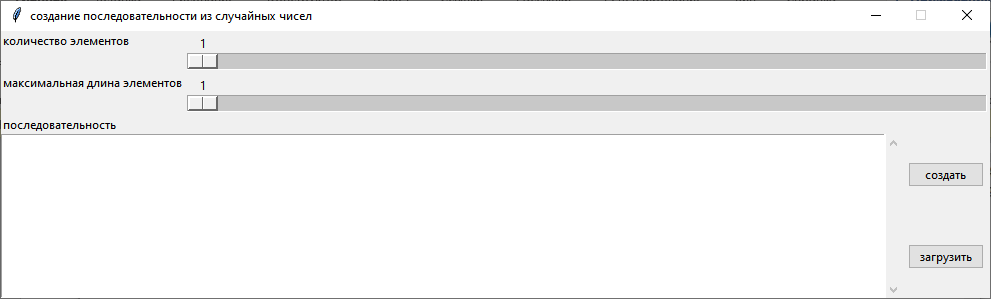


Рисунок 4 – окно создания и загрузки последовательности из случайных чисел

Видимость статуса системы можно увидеть только при загрузке очень длинной последовательности, пример представлен на рисунке 5.

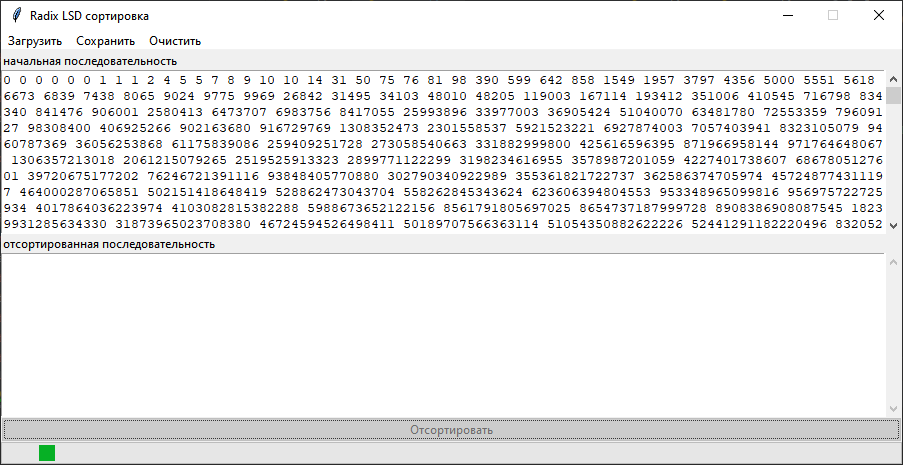


Рисунок 5 – процесс обработки очень длинного массива (худший случай алгоритмической сложности – отсортированный по порядку)

Соответствие между системой и реальным миром представлено отсутствием любых слов, значение которых было бы сложно понять. Также все подписи и элементы управления расположены вплотную друг к другу, избегая двусмысленности описания.

Согласованность представлена привычным для управления меню в шапке приложения, позволяющим выполнить дополнительные функции без длительного поиска. Также она видна в меню выбора последовательности из базы двойным нажатием.

Помощь пользователю в выявлении ошибок представлена предупреждениями о попытке выполнения неверной операции, такой как попытка загрузить последовательность без выбора.

Каждое из окон имеет минимальное количество основных элементов управления, доступных при открытии окна

# Вывод

Результат тестирования базы данных показал зависящее от количества массивов время записи и обработки данных. При использовании более мощного инструмента работы с SQL базами данных, можно было бы уменьшить это время.

# Исходный код программы

## [sort.py]

"""

модуль сортировки

"""

def sort(sorting\_sequence, place\_number=0):

"""

Shell sort

"""

sorted\_sequence = sorting\_sequence[:]

# Rearrange elements at each n/2, n/4, n/8, ... intervals

interval = len(sorted\_sequence) // 2

while interval > 0:

for i in range(interval, len(sorted\_sequence)):

temp = int(sorted\_sequence[i])

j = i

while j >= interval and int(sorted\_sequence[j - interval]) > int(temp):

sorted\_sequence[j] = sorted\_sequence[j - interval]

j -= interval

sorted\_sequence[j] = temp

interval //= 2

sorted\_sequence = list(map(str, sorted\_sequence))

return sorted\_sequence

## [base.py]

import sqlite3

import datetime

class Base:

"""

класс управления базой данных

"""

def \_\_init\_\_(self, db\_path='base.db'):

self.connection = sqlite3.connect(db\_path)

self.cursor = self.connection.cursor()

self.cursor.execute('CREATE TABLE IF NOT EXISTS sequences '

'(sequence TEXT,'

' type TEXT)')

self.connection.commit()

def add(self, sequence, type\_):

"""

процедура добавления поледовательности в базу данных

"""

assert type\_ in ('raw', 'sorted'), 'incorrect type'

if type(sequence) == str:

sequence = sequence.split()

if not all([type(element) == str for element in sequence]):

sequence = list(map(str, sequence))

self.cursor.execute('INSERT INTO sequences (sequence, type) VALUES ("{}", "{}")'

''.format(' '.join(sequence), type\_))

self.connection.commit()

def show(self, type\_):

"""

процедура получения последовательностей из базы данных

"""

self.cursor.execute(f'SELECT sequence FROM sequences WHERE type="{type\_}"')

return self.cursor.fetchall()

def clear(self):

"""

процедура удаления всех последовательностей из базы данных

"""

self.cursor.execute('DELETE FROM sequences')

self.connection.commit()

def \_\_del\_\_(self):

"""

деструктор объекта базы

"""

self.connection.close()

## [interface.py]

"""

модуль интерфейса

"""

import sort

import base

from tkinter import Tk, END, VERTICAL, IntVar, Scale, HORIZONTAL, Menu, Toplevel

from tkinter.scrolledtext import ScrolledText

from tkinter.ttk import Button, Label, Treeview, Scrollbar, Progressbar

from tkinter.messagebox import showwarning, showinfo, showerror

from tkinter.filedialog import asksaveasfilename, askopenfilename

from os import getcwd as current\_path

from os.path import exists

from random import randint

from threading import Thread

class TextEntry(ScrolledText):

"""

объект пролистываемого текстового поля с удобным интерфейсом управления для данной задачи

"""

def fill(self, text):

delete('1.0', END)

self.insert('1.0', text)

class SequenceChoice(Toplevel):

def \_\_init\_\_(self, outer\_base, parent, type\_='raw'):

"""

инициализация объекта графического интерфейса

"""

super().\_\_init\_\_(parent)

seq\_type = 'начальные' if type\_ == 'raw' else 'отсортированные' if type\_ == 'sorted' else ''

self.title(seq\_type + ' последовательности для загрузки')

self.resizable(True, False)

self.table = Treeview(self, show='headings', columns=('num', 'seq'))

self.table.column('#1', width=10)

self.table.heading('#1', text='№')

self.table.column('#2', width=800)

self.table.heading('#2', text='последовательность')

self.table.grid(row=0, column=0, sticky='new')

ysb = Scrollbar(self, orient=VERTICAL, command=self.table.yview)

self.table.configure(yscrollcommand=ysb.set)

ysb.grid(row=0, column=1, sticky='ns')

Button(self, text='загрузить', command=self.choice\_sequence).grid(row=1, column=0, sticky='new')

self.table.bind('<Double-1>', self.choice\_sequence)

# загрузка записей из базы

sequences = outer\_base.show(type\_)

if len(sequences):

for ind, sequence in enumerate(sequences, start=1):

self.table.insert("", END, values=(ind, \*sequence))

self.type = type\_

self.columnconfigure(0, weight=1)

self.grab\_set()

self.focus\_set()

def choice\_sequence(self, event=None):

"""

выбор последовательности из списка, параметр event передаётся при двойном нажатии

"""

selection = self.table.selection()

if len(selection) == 0:

showwarning(master=self, title='неправильный параметр', detail='выберите хотя бы одну строку')

elif len(selection) > 1:

showwarning(master=self, title='неправильный параметр', detail='выберите только одну строку')

else:

if self.type == 'raw':

self.master.entry.fill(self.table.item(selection[0])['values'][1])

elif self.type == 'sorted':

self.master.answer.fill(self.table.item(selection[0])['values'][1])

self.destroy()

class RandomChoice(Toplevel):

quantity\_range = 1, 1000

length\_range = 1, 150

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

"""

инициализация объекта графического интерфейса

"""

super().\_\_init\_\_(parent)

self.resizable(False, False)

self.title('создание последовательности из случайных чисел')

Label(self, text='количество элементов').grid(row=0, column=0, sticky='new')

self.quantity = IntVar(self, value=1)

Scale(self, variable=self.quantity, from\_=self.quantity\_range[0], orient=HORIZONTAL, length=800,

to=self.quantity\_range[1]).grid(row=0, column=1, sticky='new', columnspan=2)

Label(self, text='максимальная длина элементов').grid(row=1, column=0, sticky='new')

self.length = IntVar(self, value=1)

Scale(self, variable=self.length, from\_=self.quantity\_range[0], orient=HORIZONTAL, length=800,

to=self.length\_range[1]).grid(row=1, column=1, sticky='new', columnspan=2)

Label(self, text='последовательность').grid(row=2, column=0, sticky='new')

self.entry = TextEntry(self, height=10, width=110)

self.entry.grid(row=3, column=0, columnspan=2, rowspan=2, sticky='new')

Button(self, text='создать', command=self.create\_sequence).grid(row=3, column=2)

Button(self, text='загрузить', command=self.send\_to\_parent).grid(row=4, column=2)

self.grab\_set()

self.focus\_set()

def create\_sequence(self):

result\_sequence = tuple([str(randint(0, 10 \*\* randint(0, self.length.get())))

for i in range(self.quantity.get())])

self.entry.fill(' '.join(result\_sequence))

def send\_to\_parent(self):

self.master.entry.fill(self.entry.get('1.0', END))

self.destroy()

class Interface(Tk):

"""

класс с функциями управления интерфейсом

"""

def \_\_init\_\_(self):

"""

инициализация объекта графического интерфейса

"""

super().\_\_init\_\_()

self.resizable(False, False)

self.title('Shell sort / Сортировка Шелла')

Label(self, text='Начальная последовательность').grid(row=0, column=0, sticky='new')

self.entry = TextEntry(self, height=10, width=110)

self.entry.grid(row=1, column=0, columnspan=3, sticky='new')

Label(self, text='Отсортированная последовательность').grid(row=2, column=0, sticky='new')

self.answer = TextEntry(self, height=10, width=110)

self.answer.grid(row=3, column=0, columnspan=3, sticky='new')

main\_menu = Menu(self, tearoff=False)

load\_menu = Menu(main\_menu, tearoff=False)

load\_raw\_menu = Menu(load\_menu, tearoff=False)

load\_raw\_menu.add\_command(label='Из базы', command=self.load\_sequence\_from\_base)

load\_raw\_menu.add\_command(label='Из файла', command=self.load\_sequence\_from\_file)

load\_raw\_menu.add\_command(label='Случайные', command=self.load\_sequence\_random)

load\_menu.add\_cascade(label='Начальную', menu=load\_raw\_menu)

load\_sorted\_menu = Menu(load\_menu, tearoff=False)

load\_sorted\_menu.add\_command(label='Из базы', command=self.load\_sorted\_sequence\_from\_base)

load\_menu.add\_cascade(label='Отсортированную', menu=load\_sorted\_menu)

main\_menu.add\_cascade(label='Загрузить', menu=load\_menu)

save\_menu = Menu(main\_menu, tearoff=False)

raw\_seq\_menu = Menu(save\_menu, tearoff=False)

raw\_seq\_menu.add\_command(label='В базу', command=self.save\_raw\_sequence\_to\_base)

raw\_seq\_menu.add\_command(label='В файл', command=self.save\_raw\_sequence\_to\_file)

save\_menu.add\_cascade(label='Начальную', menu=raw\_seq\_menu)

sort\_seq\_menu = Menu(save\_menu, tearoff=False)

sort\_seq\_menu.add\_command(label='В базу', command=self.save\_sorted\_sequence\_to\_base)

sort\_seq\_menu.add\_command(label='В файл', command=self.save\_sorted\_sequence\_to\_file)

save\_menu.add\_cascade(label='Отсортированную', menu=sort\_seq\_menu)

main\_menu.add\_cascade(label='Сохранить', menu=save\_menu)

main\_menu.add\_command(label='Очистить', command=self.clear\_text\_entries)

self.config(menu=main\_menu)

self.start\_sort\_button = Button(self, text='Отсортировать', command=self.start\_new\_sort\_thread)

self.start\_sort\_button.grid(row=4, column=0, columnspan=3, sticky='new')

self.base = base.Base()

self.progress = Progressbar(self, mode='indeterminate')

def start\_new\_sort\_thread(self):

"""

создание потока, в котором выполняется сортировка, для поддержки работы основного окна

"""

Thread(target=self.sort\_sequence).start()

def sort\_sequence(self):

self.start\_sort\_button.config(state='disable')

self.progress.grid(row=5, column=0, columnspan=3, sticky='new')

self.progress.start()

entered\_sequence = self.entry.get('1.0', END).split()

if not len(entered\_sequence):

showerror(title='Ошибка!', message='Пустая начальная последовательность!')

return

string = ''.join(entered\_sequence)

if string.isdigit() == False:

showerror(title='Ошибка!', message='Введена некорректная последовательность! Допускаются только целочисленные числа.')

return

result\_sequence = sort.sort(entered\_sequence)

self.answer.fill(' '.join(result\_sequence))

self.progress.stop()

self.progress.grid\_forget()

self.start\_sort\_button.config(state='enable')

def load\_sequence\_from\_base(self):

self.wait\_window(SequenceChoice(self.base, self))

def load\_sequence\_from\_file(self):

path\_to\_file = askopenfilename(initialdir=current\_path(), title='выберите файл',

filetypes=(('Текстовый', '.txt'),),

defaultextension=(('Текстовый', '.txt'),))

if path\_to\_file:

if not exists(path\_to\_file):

print(f'Файл "{path\_to\_file}" не существует')

return None

with open(path\_to\_file, encoding='utf-8') as outer\_file:

# вычленение всех подстрок разделённых пробелами

entered\_sequence = outer\_file.read().strip().split()

self.entry.fill(entered\_sequence)

def load\_sequence\_random(self):

self.wait\_window(RandomChoice(self))

def save\_raw\_sequence\_to\_base(self):

sequence = self.entry.get('1.0', END)

self.base.add(sequence, 'raw')

showinfo(title='успех', message='сохранено в базу')

def save\_raw\_sequence\_to\_file(self):

sequence = self.entry.get('1.0', END)

path\_to\_file = asksaveasfilename(initialdir=current\_path(), title='выберите название',

filetypes=(('Текстовый', '.txt'),),

defaultextension=(('Текстовый', '.txt'),))

if path\_to\_file:

with open(path\_to\_file, 'w', encoding='utf-8') as outer\_file:

outer\_file.write(sequence)

showinfo(title='успех', message='сохранено в файл')

def save\_sorted\_sequence\_to\_base(self):

sequence = self.answer.get('1.0', END)

self.base.add(sequence, 'sorted')

showinfo(title='успех', message='сохранено в базу')

def save\_sorted\_sequence\_to\_file(self):

sequence = self.answer.get('1.0', END)

path\_to\_file = asksaveasfilename(initialdir=current\_path(), title='выберите название',

filetypes=(('Текстовый', '.txt'),),

defaultextension=(('Текстовый', '.txt'),))

if path\_to\_file:

with open(path\_to\_file, 'w') as outer\_file:

outer\_file.write(sequence)

showinfo(title='успех', message='сохранено в файл')

def load\_sorted\_sequence\_from\_base(self):

self.wait\_window(SequenceChoice(self.base, self, 'sorted'))

def clear\_text\_entries(self):

self.entry.fill('')

self.answer.fill('')

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

root = Interface()

root.mainloop()

## [ test load.py ]

import os

import base

import sort

from random import randint

import time

import traceback

from os import remove

def add\_to\_(base\_object, sequence\_number, max\_elements\_length=2, max\_elements\_number=5):

for sequence\_iter in range(sequence\_number):

elements\_length, elements\_number = randint(1, max\_elements\_length), randint(1, max\_elements\_number)

sequence = tuple([str(randint(0, 10 \*\* randint(0, elements\_length))) for i in range(elements\_number)])

base\_object.add(sequence, 'raw')

def sort\_sequences\_of\_(base\_object):

for sequence in base\_object.show('raw'):

sort.sort(sequence)

def clear\_(base\_object):

base\_object.clear()

b = base.Base('test.db')

b.clear()

for number in (100, 1000, 10000):

print('-' \* 100)

print('test load {}'.format(number))

try:

print('load:\n')

start = time.time()

add\_to\_(b, number)

load\_time = time.time() - start

print('\tsuccess\n')

print('\ttotal:\t\t{}\n\tper iter:\t{}'.format(load\_time, load\_time / number))

except Exception:

print('\tfail\n')

traceback.print\_exc()

try:

print('sort:\n')

sort\_time = [0, 0, 0]

start = time.time()

for i in range(3):

start = time.time()

sort\_sequences\_of\_(b)

sort\_time[0] += (time.time() - start)

avg\_sort\_time = sort\_time[0] / 3

print('\tsuccess\n')

print('\ttotal:\t\t{}\n\tper iter:\t{}'.format(avg\_sort\_time, avg\_sort\_time / number))

except Exception:

print('\tfail\n')

traceback.print\_exc()

try:

print('clear:\n')

start = time.time()

clear\_(b)

clear\_time = time.time() - start

print('\tsuccess\n')

print('\ttotal:\t\t{}'.format(clear\_time))

except Exception:

print('\tfail\n')

traceback.print\_exc()

os.system("PAUSE")

del b

remove('test.db')