Дипломная работа на тему:

Создание GUI-приложения для преобразования графических файлов в японскую нонограмму на языке программирования “Python”.

Содержание:

Введение.

Японская но**н**ограмма (не путать с математическим термином «но**м**ограмма») - головоломка, похожая внешне на кроссворд, в которой закодированы не слова, а изображение. Последние закодированы числами, расположенными слева от строк, а также сверху над столбцами. Количество чисел показывает, сколько групп чёрных (либо своего цвета, для цветных кроссвордов) клеток находятся в соответствующих строке или столбце, а сами числа — сколько слитных клеток содержит каждая из этих групп (например, набор из трёх чисел — 4, 1, и 3 означает, что в этом ряду есть три группы: первая — из четырёх, вторая — из одной, третья — из трёх чёрных клеток). Группы должны быть разделены, как минимум, одной пустой клеткой; необходимо определить размещение групп клеток (рис. 1).

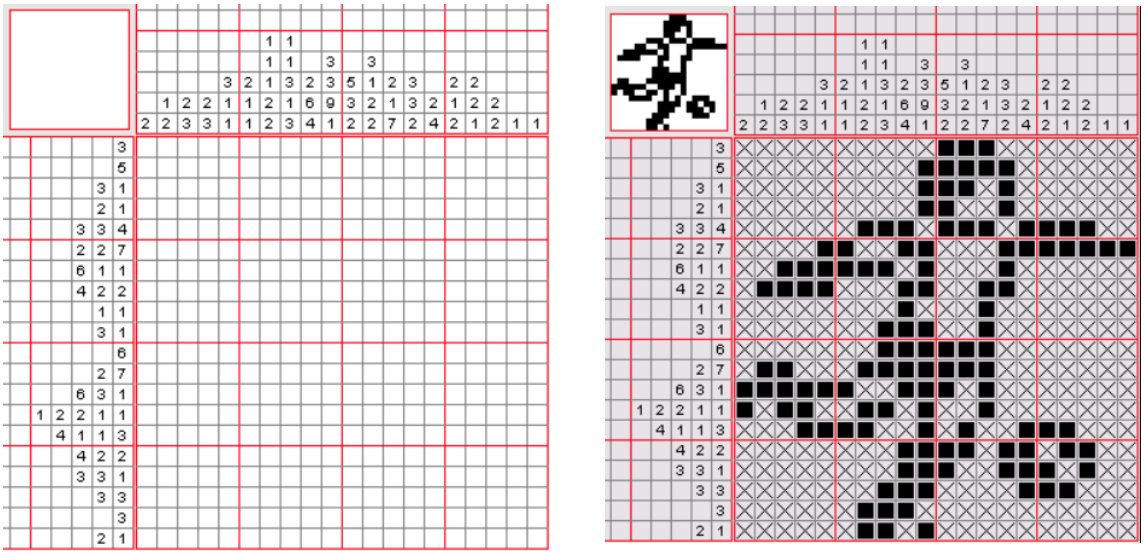


Рисунок 1 - пример нонограммы. Слева — исходная задача; справа - решение

Японские головоломки появились в Японии в конце XX века и широкую известность получили в 1989 -1990 годах. Вскоре о номограммах узнали и в России, здесь они получили название «японские кроссворды» или «японские головоломки».

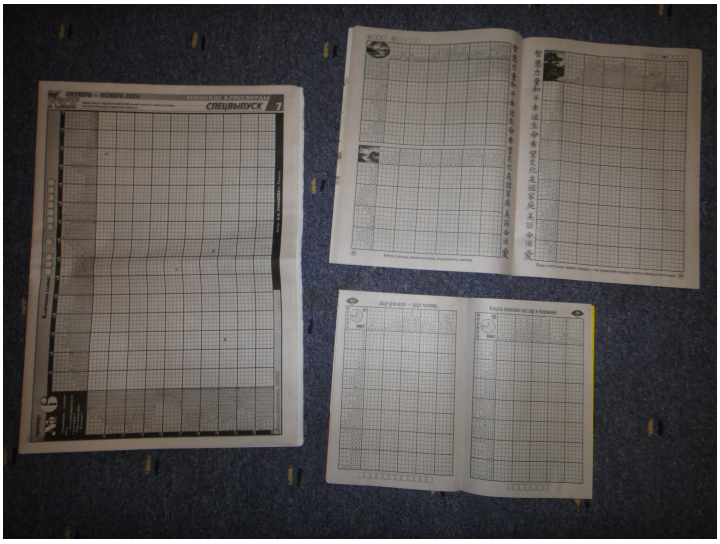
Основные требования к составлению нонограммы:

- правильно составленная головоломка должна иметь только одно решение.

- фоновый цвет (цвет пустых клеток игрового поля) японского кроссворда должен быть белым.

- недопустимо наличие строк и столбцов, в которых «нечего разгадывать» (обычно под этим подразумевают отсутствие одинарных пустых клеток между группами цифр разных цветов, и отсутствие двойных пустых клеток между группами одного цвета. В этом случае группы цифр имеют единственный вариант расположения, так как они занимают все свободное пространство). **[1]**

Мне интересна тема дипломной работы, поскольку я сам, будучи подростком в начале 2000-х годов, увлекался этим. Проживал тогда я в городе. Новосибирске, где в газетных киосках продавали кроссворды под названием *«Фудзияма»* (это такой священный вулкан). Собственно, это и будет название моей программы, ничего лучше для названия я уже не придумаю.



В настоящее время японские кроссворды до сих пор есть в продаже, что для меня даже было несколько неожиданно. Например, специально для подготовки дипломной работы уже здесь, в Москве, я приобрёл экземпляры газет *«Крот»*, *«100 самураев»* и *«Сакура»* (все - на фото).

Наверняка, издательства не составляют кроссворды вручную, а так же пользуются определёнными скриптами, и мой код тоже вполне может претендовать на то или иное коммерческое использование, поскольку выполняет базовую задачу. На мой взгляд, подобный «софт» можно коммерциализировать тремя способами:

1) Самое простое - продать авторские права и исходный код, не тратя больше своё время и нервы на общение с заказчиком, поскольку дальше с кодом будут работать уже его сотрудники-программисты.

2) Продать только скомпилированный под Windows / Linux файл, и осуществлять его поддержку по требованиям заказчика (процесс создания скомпилированной программы тоже будет описан в этой дипломной работе). Дополнительно для защиты от стороннего использования, можно сделать всплывающее окно, где потребуется ввести ключ к доступу к программе.

3) Создать telegram-бота, который будет доступен по всему миру, и набирать подписчиков. И чем больше людей им пользуется, тем, скорее всего, больше рекламы можно будет разместить, и зарабатывать на этом. Правда, для этого нужно написать, собственно, сам код бота, и сделать в боте интерфейс программы на разных языках (выбирать в отдельном меню и обрабатывать «хендлером»).

Таким образом, тема дипломной работы имеет актуальность на сегодняшний день, и вполне имеет потенциал коммерческой реализации.

Далее будет описание работы моего кода на конкретном примере.

1. Создание проекта

1.1 Обзор используемых в работе библиотек.

В своей работе я использую несколько специализированных библиотек:

Tkinter - это стандартная библиотека Python для создания графического интерфейса пользователя (GUI). Она позволяет разработчикам легко создавать окна, кнопки, текстовые поля и другие элементы интерфейса. Основные особенности Tkinter включают:

- Простота использования: Легкий синтаксис и интуитивно понятный интерфейс.
- Кросс-платформенность: Работает на Windows, macOS и Linux.
- Стандартная библиотека: Входит в стандартный набор библиотек Python, поэтому не требует дополнительной установки.
- Разнообразие виджетов: Поддерживает множество стандартных элементов управления, таких как метки, кнопки, текстовые области и др.

Tkinter часто используется для быстрого прототипирования GUI-приложений и обучения основам создания интерфейсов. **[2]**

Pillow - это форк (отдельная ветка разработки) библиотеки PIL (Python Imaging Library), используемый для работы с изображениями в Python. Pillow предоставляет широкие возможности для обработки изображений, включая загрузку, сохранение, редактирование и конвертацию форматов. Основные особенности Pillow включают:

- Поддержка множества форматов: Работа с JPEG, PNG, GIF, TIFF и многими другими форматами изображений.
- Редактирование изображений: Изменение размера, обрезка, поворот, применение фильтров и эффектов.
- Создание изображений: Генерация новых изображений программным путем.
- Кросс-платформенность: Работает на Windows, macOS и Linux.
- Активное сообщество: Постоянные обновления и поддержка от сообщества разработчиков.

Pillow часто используется в проектах, связанных с обработкой изображений, веб-разработке и создании приложений с графическим контентом. **[3]**

NumPy (Numeric Python) — это популярная библиотека для Python, предназначенная для работы с многомерными массивами и матрицами. Она предоставляет мощные инструменты для выполнения математических операций, таких как линейная алгебра, статистика и преобразования Фурье. Основные особенности NumPy включают:

- Высокопроизводительные многомерные массивы: Быстрое выполнение операций над большими объемами данных.

- Векторизация: Применение операций ко всем элементам массива одновременно, что ускоряет выполнение кода.

- Интеграция с другими библиотеками: Совместима с Pandas, SciPy, Matplotlib и другими научными библиотеками.

- Удобство работы с данными: Поддержка чтения/записи данных в разные форматы файлов.

NumPy широко используется в научных исследованиях, машинном обучении, обработке данных и других областях, требующих интенсивных вычислений. **[4]**

Matplotlib - это библиотека Python для создания статичных, анимированных и интерактивных визуализаций данных. Она предоставляет разнообразные инструменты для построения графиков, диаграмм, гистограмм и других типов визуальных представлений данных. Основные особенности Matplotlib включают:

- Гибкость: Поддержка различных типов графиков и диаграмм.

- Настраиваемость: Возможность настройки внешнего вида графиков вплоть до мельчайших деталей.

- Интеграция с другими библиотеками: Совместимость с NumPy, Pandas и другими научными библиотеками.

- Кросс-платформенность: Работает на Windows, macOS и Linux.

- Документированность: Подробная документация и множество примеров использования.

Matplotlib широко используется в научных исследованиях, анализе данных и машинном обучении для визуализации результатов и представления информации в удобном виде. **[5]**

1.2 Настройка IDE

Свой дипломную работу я буду выполнять в специализированной IDE “Pycharm” версии 2024.2.4 с использованием стабильной версии Python 3.12 и в отдельном виртуальном окружении. Вот список библиотек и их версий из файла requirements.txt.

```
contourpy==1.3.0
cycler==0.12.1
fonttools==4.54.1
kiwisolver==1.4.7
matplotlib==3.9.2
numpy==2.1.3
packaging==24.1
pillow==11.0.0
pyparsing==3.2.0
python-dateutil==2.9.0.post0
six==1.16.0
```

2. Описание логики работы программы

2.1. Создание графического интерфейса в библиотеке tkinter

В ПРОЦЕССЕ ОФОРМЛЕНИЯ...

2.2. Работа с изображением в библиотеке pillow

В ПРОЦЕССЕ ОФОРМЛЕНИЯ...

2.3. Бинаризация (трансформация пикселей в матрицу чисел) в библиотеке numpy

В ПРОЦЕССЕ ОФОРМЛЕНИЯ...

2.4. Графическое представление номограммы в matplotlib, экспорт готового файла

В ПРОЦЕССЕ ОФОРМЛЕНИЯ...

3. Демонстрация работы кода на конкретном примере

В ПРОЦЕССЕ ОФОРМЛЕНИЯ...

Заключение.

Как было показано на примере, код выполняет свою основную задачу, однако есть возможности по его улучшению и расширению функционала, а именно, можно впоследствии сделать следующее:

- перейти с tkinter на другую библиотеку по созданию приложений (DearPygui, PyQT, wxPython или другие), поскольку tkinter имеет на сегодняшний день довольно устаревший интерфейс.
- добавить в окно программы дополнительные возможности для пользователя: предпросмотр изображений, из масштабирование, автоматическую обрезку пустого места по краям, выбор пути для экспорта файла и другое.
- добавить возможность преобразования рисунка в цветную нонограмму. Для этого нужно дополнительное поле с выбором количества цветов, а также корректировка кода, поскольку в цветной номограмме пробелов (пустых клеток) между цветами может и не быть.
- добавить интерактивный игровой функционал, когда изображение будет случайно выбираться из определённой базы, и пользователь сможет самостоятельно закрашивать клетки, решая головоломку в отдельном окне, например, нажимая кнопки мыши.
- создание того самого telegram-бота для одного из возможных способов коммерческой реализации программы и доступа к ней со всего мира.

Итак, нонограммы до сих пор популярны как в печатном виде, так и в виде компьютерных игр и мобильных приложений. Это отличный способ тренировать логику и внимательность, а также просто наслаждаться процессом решения загадок и созданием изображений!

Приложение 1. Полный код программы.

```
import tkinter as tk
from tkinter import filedialog, messagebox

import matplotlib.pyplot as plt
import numpy as np
from PIL import ImageTk, Image


def select():
    global selected_file_path
    selected_file_path = filedialog.askopenfilename(title="Выберите изображение",
                                            filetypes=[
                                                ("Только", ("*.jpg", "*.jpeg", "*.png", "*.gif", "*.bmp"))])
    if selected_file_path:
        metadata_text_field.delete(1.0, tk.END)
        metadata = {}
        width, height = Image.open(selected_file_path).size

        metadata['Выбрано'] = str(selected_file_path)
        metadata['Размер '] = f"{width} x {height}"

        for key, value in metadata.items():
            metadata_text_field.insert(tk.END, f"{key}: {value}\n")

        if width > 80 or height > 80:
            if width >= height:
                messagebox.showwarning("Предупреждение",
                                       f"Размер файла слишком большой; нонограмма "
                                       f"будет уменьшена до {80} x {int(80 * height / width)}")
            if width < height:
                messagebox.showwarning("Предупреждение",
                                       f"Размер файла слишком большой; нонограмма "
                                       f"будет уменьшена до {int(80 * width / height)} x {80}")
    select_button2.configure(state=tk.ACTIVE)
    return selected_file_path


def export():
    binary_img = image_to_array(selected_file_path, int(slider.get()))
    row_hints, col_hints = generate_hints(binary_img)
    if selected_option.get() == 'Только решение':
        draw_nonogram(row_hints, col_hints, binary_img, 'gray', output_full_pdf)
    elif selected_option.get() == 'Только задача':
        arr = binary_img
        arr[:] = 1
        draw_nonogram(row_hints, col_hints, arr, 'binary', output_empty_pdf)
    else:
        draw_nonogram(row_hints, col_hints, binary_img, 'gray', output_full_pdf)
        arr = binary_img
        arr[:] = 1
        draw_nonogram(row_hints, col_hints, arr, 'binary', output_empty_pdf)


def image_to_array(selected_file_path, threshold):
    img = Image.open(selected_file_path).convert('L')
    width, height = img.size
    if width > 80 or height > 80:
        max_size = (80, 80)
        img.thumbnail(max_size)
    img = np.array(img)
    if not inverse.get():
        binary_img = (img > threshold).astype(int)
```

```
    else:
        binary_img = (img < threshold).astype(int)
    return binary_img


def generate_hints(binary_img):
    row_hints = []
    col_hints = []

    for row in binary_img:
        hint = []
        count = 0
        for pixel in row:
            if pixel == 0:
                count += 1
            else:
                if count > 0:
                    hint.append(count)
                count = 0
        if count > 0:
            hint.append(count)
        row_hints.append(hint if hint else [0])

    for col in binary_img.T:
        hint = []
        count = 0
        for pixel in col:
            if pixel == 0:
                count += 1
            else:
                if count > 0:
                    hint.append(count)
                count = 0
        if count > 0:
            hint.append(count)
        col_hints.append(hint if hint else [0])
    return row_hints, col_hints


def draw_nonogram(row_hints, col_hints, binary_img, cmap, export_pdf):
    fig, ax = plt.subplots()
    grid_height = len(row_hints)
    grid_width = len(col_hints)
    ax.imshow(binary_img, cmap=cmap)
    ax.set_xticks(np.arange(-0.5, grid_width, 1), minor=True)
    ax.set_yticks(np.arange(-0.5, grid_height, 1), minor=True)
    ax.grid(which="minor", color="gray", linestyle='-', linewidth=0.5)
    ax.tick_params(which="both", bottom=False, left=False, labelbottom=False, labelleft=False)
    for i, hint_row in enumerate(row_hints):
        ax.text(-1, i, f"{'  '.join(map(str, hint_row))}", va='center', ha='right', fontsize=2)
    for j, hint_col in enumerate(col_hints):
        ax.text(j, -1, f"{'\n'.join(map(str, hint_col))}", va='bottom', ha='center', fontsize=2)
    fig.tight_layout()
    plt.savefig(export_pdf, format='pdf', bbox_inches='tight')


root = tk.Tk()
root.title("ФУДЗИЯМА - создание японских кроссвордов")
root.geometry("640x360+640+360")
root.iconbitmap("Fuji_s.ico")
root.resizable(False, False)
background_image = Image.open("Fuji_s.jpg")
background_photo = ImageTk.PhotoImage(background_image)
background_label = tk.Label(root, image=str(background_photo))
background_label.place(x=0, y=0, relwidth=1, relheight=1)
```

```
select_button1 = tk.Button(root, text=" Выбрать файл ", command=select)
select_button1.place(x=25, y=200)
select_button2 = tk.Button(root, state=tk.DISABLED, text="       Экспорт       ", command=export)
select_button2.place(x=25, y=235)
metadata_text_field = tk.Text(root, height=4, width=60)
metadata_text_field.place(x=130, y=200)
slider = tk.Scale(root, from_=1, to=255, orient='horizontal', length=300,
                  label='Порог преобразования (бинаризации):', variable=tk.IntVar(value=128))
slider.place(x=25, y=298)
label_name = 'Настройки экспорта:'
label = tk.Label(root, text=f"{label_name:^174}")
label.place(x=25, y=273)
selected_option = tk.StringVar()
selected_option.set('Только задача')
option1 = tk.Radiobutton(root, text='Только задача', variable=selected_option, value='Только задача')
option2 = tk.Radiobutton(root, text='Только решение', variable=selected_option, value='Только решение')
option3 = tk.Radiobutton(root, text='Обе нонограммы', variable=selected_option, value='Обе нонограммы')
option1.place(x=355, y=300)
option2.place(x=355, y=330)
option3.place(x=486, y=300)
inverse = tk.BooleanVar()
checkbox = tk.Checkbutton(root, text="Инвертировать", variable=inverse)
checkbox.place(x=500, y=330)
selected_file_path = ""
output_full_pdf = "SOLVE.pdf"
output_empty_pdf = "TASK.pdf"

root.mainloop()
```

Приложение 2. Список источников.

1. https://ru.wikipedia.org/wiki/Японский_кроссворд
2. https://docs.python.org/3/library/tkinter.html
3. https://pillow.readthedocs.io/en/stable/handbook/index.html
4. https://numpy.org/doc/stable/user/index.html
5. https://matplotlib.org/stable/users/index.html