ЗВІТ

про виконання лабораторної роботи № 3

«Розробка ASCII ART генератора для візуалізації текстових даних»

з дисципліни

«Спеціалізовані мови програмування»

студенки групи РІ-31

Заяць Анастасії Назарівної

**Мета:** Cтворення додатка Генератора ASCII-арту.

**Умова завдання:**

Завдання 1: Введення користувача

Створіть Python-програму, яка приймає введення користувача для слова або фрази, яку треба перетворити в ASCII-арт.

Завдання 2: Бібліотека ASCII-арту

Інтегруйте бібліотеку ASCII-арту (наприклад, pyfiglet або art) у вашу програму для генерації ASCII-арту з введення користувача

Завдання 3: Вибір шрифту

Дозвольте користувачам вибирати різні стилі шрифтів для свого ASCII-арту. Надайте список доступних шрифтів та дозвольте їм вибрати один.

Завдання 4: Колір тексту

Реалізуйте опцію вибору користувачем кольору тексту для їхнього ASCII-арту. Підтримуйте основний вибір кольорів (наприклад, червоний, синій, зелений).

Завдання 5: Форматування виводу

Переконайтеся, що створений ASCII-арт правильно відформатований та вирівнюється на екрані для зручності читання.

Завдання 6: Збереження у файл

Додайте функціональність для збереження створеного ASCII-арту у текстовому файлі, щоб користувачі могли легко завантажувати та обмінюватися своїми творіннями.

Завдання 7: Розмір ARTу

Дозвольте користувачам вказувати розмір (ширина і висота) ASCII-арту, який вони хочуть створити. Масштабуйте текст відповідно.

Завдання 8: Вибір символів

Дозвольте користувачам вибирати символи, які вони хочуть використовувати для створення ASCII-арту (наприклад, '@', '#', '\*', тощо).

Завдання 9: Функція попереднього перегляду

Реалізуйте функцію попереднього перегляду, яка показує користувачам попередній перегляд їхнього ASCII-арту перед остаточним збереженням.

Завдання 10: Інтерфейс, зрозумілий для користувача

Створіть зручний для користувача інтерфейс командного рядку для додатка, щоб зробити його інтуїтивно зрозумілим та легким у використанні.

**Текст програми:**

import os

import pyfiglet

from BLL.constants import ansi\_colors

class AsciiArtGenerator:

    def \_\_init\_\_(self):

        self.art\_text = ""

        self.font = "standard"

        self.color = "white"

        self.width\_factor = 1

        self.height\_factor = 1

        self.art\_symbol = ""

        self.ascii\_text = ""

    def get\_input(self):

        while True:

            text = input("Enter the text you want to convert to ASCII art: ").strip()

            if text:

                self.art\_text = text

                break

            else:

                print("Input cannot be empty. Please try again.")

    def get\_font(self):

        example\_fonts = ['standard', 'slant', 'block', 'bubble', 'digital']

        print("Available fonts: ", ', '.join(example\_fonts))

        while True:

            font = input("Select a font (or press Enter for 'standard'): ").strip()

            if not font:

                self.font = "standard"

                break

            elif font in example\_fonts:

                self.font = font

                break

            else:

                print("Invalid font. Try again.")

    def get\_color(self):

        example\_colors = ['red', 'green', 'yellow', 'blue', 'magenta', 'cyan', 'white']

        print("Available colors: ", ', '.join(example\_colors))

        while True:

            color = input("Select a color (or press Enter for 'white'): ").strip()

            if not color:

                self.color = "white"

                break

            elif color in example\_colors:

                self.color = color

                break

            else:

                print("Invalid color. Try again.")

    def get\_scaling\_factors(self):

        while True:

            try:

                width\_factor = input("Enter the width scaling factor (default is 1): ").strip()

                height\_factor = input("Enter the height scaling factor (default is 1): ").strip()

                self.width\_factor = int(width\_factor) if width\_factor else 1

                self.height\_factor = int(height\_factor) if height\_factor else 1

                if self.width\_factor > 0 and self.height\_factor > 0:

                    break

                else:

                    print("Scaling factors must be positive integers. Try again.")

            except ValueError:

                print("Invalid input. Please enter valid integers for scaling factors.")

    def get\_symbol(self):

        while True:

            symbol = input("Enter the symbol you want to use in the ASCII art ('@', '#', '\*',...): ").strip()

            if symbol:

                self.art\_symbol = symbol

                break

            else:

                print("Using default symbol '#' for the ASCII art.")

                self.art\_symbol = "#"

    def scale\_ascii\_art(self, ascii\_art):

        scaled\_lines = []

        for line in ascii\_art.splitlines():

            scaled\_line = "".join(char \* self.width\_factor for char in line)

            for \_ in range(self.height\_factor):

                scaled\_lines.append(scaled\_line)

        return "\n".join(scaled\_lines)

    def generate\_art(self):

        try:

            ascii\_art = pyfiglet.figlet\_format(self.art\_text, font=self.font)

            scaled\_art = self.scale\_ascii\_art(ascii\_art)

            color\_code = ansi\_colors.ANSI\_COLORS.get(self.color, '\033[37m')

            colored\_art = f"{color\_code}{scaled\_art}\033[0m"

            self.ascii\_text = colored\_art

            return colored\_art

        except Exception as e:

            print(f"Error generating ASCII art: {e}")

            return None

    def generate\_art\_symbol(self):

        try:

            ascii\_art = pyfiglet.figlet\_format(self.art\_text, "banner3")

            scaled\_art = self.scale\_ascii\_art(ascii\_art)

            color\_code = ansi\_colors.ANSI\_COLORS.get(self.color, '\033[37m')

            colored\_art = f"{color\_code}{scaled\_art}\033[0m"

            ascii\_art = colored\_art.replace("#", self.art\_symbol)

            self.ascii\_text = ascii\_art

            print(ascii\_art)

            return ascii\_art

        except Exception as e:

            print(f"Error generating ASCII art: {e}")

            return None

    def preview\_art(self):

        art\_preview = self.generate\_art()

        if art\_preview:

            print("Preview of your ASCII art:")

            print(art\_preview)

    def save\_to\_file(self):

        try:

            folder\_to\_save = os.path.abspath(os.path.join(os.getcwd(), os.pardir, "calculator", "Sources"))

            os.makedirs(folder\_to\_save, exist\_ok=True)

            file\_name = input("Enter the file name to save the ASCII art (e.g., art): ").strip()

            formatted\_file\_name = os.path.join(folder\_to\_save, f"{file\_name}.txt")

            ascii\_art = self.ascii\_text

            with open(formatted\_file\_name, 'w') as file:

                file.write(ascii\_art.replace('\033[0m', ''))

            print(f"ASCII art saved to {formatted\_file\_name}.")

        except Exception as e:

            print(f"Error saving ASCII art to file: {e}")

    def run(self):

        while True:

            self.get\_input()

            self.get\_font()

            self.get\_color()

            self.get\_scaling\_factors()

            self.preview\_art()

            save\_choice = input("Do you want to save the ASCII art to a file? (yes/no): ").strip().lower()

            if save\_choice == 'yes':

                self.save\_to\_file()

            self.get\_symbol()

            self.generate\_art\_symbol()

            save\_choice = input("Do you want to save the ASCII art to a file? (yes/no): ").strip().lower()

            if save\_choice == 'yes':

                self.save\_to\_file()

            if input('Do you want to create another ASCII art? (yes/no): ').lower() != 'yes':

                print("Thank you for using the ASCII Art Generator!")

                break

**Висновки:** Виконуючи ці завдання, студенти створять універсальний Генератор ASCII-арту, який дозволить користувачам налаштовувати свої творіння з різними шрифтами, кольорами, розмірами та символами. Проект надасть практичний досвід роботи з введенням користувача, зовнішніми бібліотеками, роботою з файлами та дизайном інтерфейсу користувача в Python.