ЗВІТ

про виконання лабораторної роботи № 4

«Розробка ASCII ART генератора для візуалізації 2D-фігур»

з дисципліни

«Спеціалізовані мови програмування»

студентки групи РІ-31

Заяць Анастасії Назарівної

**Мета:** Створення Генератора ASCII-арту без використання зовнішніх бібліотек

**Умова завдання:**

Завдання 1: Введення користувача

Створіть програму Python, яка отримує введення користувача щодо слова або фрази, яку вони хочуть перетворити в ASCII-арт.

Завдання 2: Набір символів

Визначте набір символів (наприклад, '@', '#', '\*', тощо), які будуть використовуватися для створення ASCII-арту. Ці символи будуть відображати різні відтінки.

Завдання 3: Розміри Art-у

Запитайте у користувача розміри (ширина і висота) ASCII-арту, який вони хочуть створити. Переконайтеся, що розміри в межах керованого діапазону

Завдання 4: Функція генерації Art-у

Напишіть функцію, яка генерує ASCII-арт на основі введення користувача, набору символів та розмірів. Використовуйте введення користувача, щоб визначити, які символи використовувати для кожної позиції в Art-у.

Завдання 5: Вирівнювання тексту

Реалізуйте опції вирівнювання тексту (ліво, центр, право), щоб користувачі могли вибирати, як їх ASCII-арт розміщується на екрані.

Завдання 6: Відображення мистецтва

Відобразіть створений ASCII-арт на екрані за допомогою стандартних функцій друку Python.

Завдання 7: Збереження у файл

Додайте можливість зберігати створений ASCII-арт у текстовий файл, щоб користувачі могли легко завантажувати та обмінюватися своїми творіннями.

Завдання 8: Варіанти кольорів

Дозвольте користувачам вибирати опції кольорів (чорно-білий, відтінки сірого) для свого ASCII-арту.

Завдання 9: Функція попереднього перегляду

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

Завдання 10: Інтерфейс, зрозумілий для користувача

Створіть інтерфейс для користувача у командному рядку, щоб зробити програму легкою та інтуїтивно зрозумілою для використання.

**Текст програми:**

import os

from BLL.constants import ansi\_colors

from Sources.ABC123hash import draw\_char\_hash

from Sources.ABC123dog import draw\_char\_dog

from Sources.ABC123asterism import draw\_char\_asterism

class MyAsciiArtGenerator:

    def \_\_init\_\_(self):

        self.art\_text = ""

        self.color = "white"

        self.width\_factor = 1

        self.height\_factor = 1

        self.art\_symbol = ""

        self.ascii\_text = ""

        self.max\_width = 140

        self.max\_height = 28

        self.alignment = "center"

    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.upper()

                break

            else:

                print("Input cannot be empty. Please 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 get\_alignment(self):

        while True:

            alignment = input("Choose text alignment (left, center, right): ").strip().lower()

            if alignment in ['left', 'center', 'right']:

                self.alignment = alignment

                break

            else:

                print("Invalid input. Please choose 'left', 'center', or 'right'.")

    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 figlet\_format(self, art\_text, font):

            result = [""] \* 7

            for letter in art\_text:

                if font == "#":

                    letter\_art = draw\_char\_hash(letter)

                elif font == "@":

                    letter\_art = draw\_char\_dog(letter)

                elif font == "\*":

                    letter\_art = draw\_char\_asterism(letter)

                else:

                    raise ValueError("Invalid font symbol. Choose '#', '@', or '\*'.")

                for i in range(7):

                    result[i] += letter\_art[i]

            return "\n".join(result)

            # for line in result:

            #     print(line)

            # print()

    def generate\_art\_symbol(self):

        try:

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

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

            art\_lines = scaled\_art.splitlines()

            aligned\_art\_lines = []

            for line in art\_lines:

                if self.alignment == 'center':

                    aligned\_line = line.center(self.max\_width)

                elif self.alignment == 'right':

                    aligned\_line = line.rjust(self.max\_width)

                else:  # self.alignment == 'left'

                    aligned\_line = line.ljust(self.max\_width)

                aligned\_art\_lines.append(aligned\_line)

            aligned\_art = "\n".join(aligned\_art\_lines)

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

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

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

            self.ascii\_text = ascii\_art

            canvas\_width, canvas\_height = self.get\_canvas\_size()

            canvas = self.create\_canvas(canvas\_width, canvas\_height)

            art\_lines = ascii\_art.splitlines()

            for i in range(min(canvas\_height, len(art\_lines))):

                for j in range(min(canvas\_width, len(art\_lines[i]))):

                    canvas[i][j] = art\_lines[i][j]

            # Виводимо результат

            for line in canvas:

                print(''.join(line))

            return canvas

        except Exception as e:

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

            return None

    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 align\_text(art, alignment='center', width=80):

        lines = art.split('\n')

        aligned\_art = ""

        for line in lines:

            if alignment == 'center':

                aligned\_art += line.center(width) + '\n'

            elif alignment == 'left':

                aligned\_art += line.ljust(width) + '\n'

            elif alignment == 'right':

                aligned\_art += line.rjust(width) + '\n'

        return aligned\_art

    def get\_canvas\_size(self):

        while True:

            try:

                width = int(input("Enter the width of the canvas (max 140): ").strip())

                height = int(input("Enter the height of the canvas (max 28): ").strip())

                if 1 <= width <= 140 and 1 <= height <= 28:

                    return width, height

                else:

                    print("Width must be between 1 and 140, height must be between 1 and 28. Please try again.")

            except ValueError:

                print("Invalid input. Please enter valid integers for width and height.")

    def create\_canvas(self, width, height):

        return [[' ' for \_ in range(width)] for \_ in range(height)]

    def run(self):

        while True:

            self.get\_input()

            self.get\_symbol()

            self.get\_color()

            self.get\_scaling\_factors()

            self.get\_alignment()

            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 я створила генератор ASCII-арту з нуля, та надати можливість налаштовувати символи, розміри, вирівнювання та кольори, що дозволить їм глибше розібратися як створюється ASCII-арт