МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра ІСМ



ЗВІТ

Про виконання лабораторної роботи №4

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

З дисципліни

«Спеціалізовані мови програмування»

Студента групи РІ – 31

Тураша Івана Павловича

Прийняв викладач

Щербак С.С.

**Лабораторна робота № 4. Розробка ASCII ART генератора для візуалізації 2D-фігур**

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

**План роботи**

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.

**Основний код програми:**

import random

import string

from Shared.classes.validators import Validators

from Shared.classes.incorrect\_character\_exception import IncorrectCharacterException

class Ascii:

    def \_\_init\_\_(self, text, width = 0, height = 0, color = "\033[39m", shadow = "#", text\_s = "#", highlight = "#", justify = "left"):

        self.text = text

        self.shadow = shadow

        self.text\_s = text\_s

        self.highlight = highlight

        self.height = height

        self.width = Validators.verify\_width(width)

        self.color = "\033[" + str(random.randint(31, 39)) + "m" if color == "random" else color

        self.color\_reset = "\033[0m"

        self.justify = justify

        self.font = self.\_\_load\_font()

    def print(self):

        art = self.\_\_format\_art()

        print(self.color + art + self.color\_reset)

        return art

    def \_\_wrap\_art(self):

        wrapped\_text = []

        length = 0

        current\_line = ""

        for char in self.text.upper():

            if char in ["@", "#"]:

                length += 9

            if char in ["M", "W", "4", "\*"]:

                length += 8

            elif char in self.font:

                length += 7

            else:

                raise IncorrectCharacterException("The character " + char + " is not a valid character.")

            if length > self.width:

                wrapped\_text.append(current\_line)

                current\_line = char

                length = 8 if char in ["M", "W", "4"] else 7

            else:

                current\_line += char

        if current\_line:

            wrapped\_text.append(current\_line)

        return wrapped\_text

    def \_\_format\_art(self):

        wrapped\_art = self.\_\_wrap\_art()

        art\_lines = []

        for chunk in wrapped\_art:

            unsorted\_art\_list = []

            for art\_char in chunk:

                font\_art = self.font[art\_char.upper()]

                formatted\_font\_art = (font\_art.replace('\*', self.highlight)

                                      .replace('#', self.text\_s).replace('&', self.shadow))

                split\_font\_art = formatted\_font\_art.splitlines()

                unsorted\_art\_list.append(split\_font\_art)

            art\_list = []

            for row in zip(\*unsorted\_art\_list):

                row\_str = "".join(row)

                match self.justify:

                    case "left":

                        art\_list.append(row\_str)

                    case "center":

                        width = self.width - len(row\_str)

                        padding = " " \* (width // 2)

                        art\_list.append(padding + row\_str + padding)

                    case "right":

                        width = self.width - len(row\_str)

                        padding = " " \* width

                        art\_list.append(padding + row\_str)

            art\_lines.append("\n".join(art\_list))

        art = "\n\n".join(art\_lines)

        art\_height = len(art.splitlines())

        height\_diff = self.height - art\_height

        padding = "\n" \* (height\_diff // 2) if height\_diff > 0 else ""

        return padding + art + padding

    @staticmethod

    def \_\_load\_font():

        keys = (list(string.ascii\_uppercase) + list(string.digits) +

                ["!", "@", "#", "$", "%", "^", "&", "\*", "(", ")", "-", "\_", "=",

                 "+", "[", "]", ";", ":", "'", '"', ",", ".", "/", "<", ">", "?", " "])

        font = {}

        with open("Sources/font.txt", "r") as file:

            i = 0

            for line in file:

                if line.strip() == "$":

                    i+= 1

                elif i < len(keys):

                    key = keys[i]

                    font[key] = font.get(key, "") + line

        return font

**Вмсновок:** Виконуючи цю лабораторну роботу,я створив генератор ASCII-арту з нуля, та надала можливість налаштовувати символи, розміри, вирівнювання та кольори, що дозволило мені глибше розібратися як створюється ASCII-арт.