Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

про виконання лабораторної роботи № 4

з курсу:

«Спеціалізовані мови програмування»

**Виконала:**

студентка гр. ІТ-31

Ірина ПЕРХУН

**Прийняв:**

Сергій ЩЕРБАК

Львів 2023

**Тема:** Розробка ASCII ART генератора для візуалізації 2D-фігур

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

**Хід 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.*

**ascii\_art\_generator.py**

class ASCIIArtGenerator:

    """

    Class for generating and manipulating colored ASCII art.

    Attributes:

        letters (dict): Dictionary containing ASCII representations of letters.

    Methods:

        generate\_colored\_ascii\_art(text, letters, width, height, alignment="left"):

            Generate colored ASCII art from the given text.

        save\_ascii\_art\_to\_file(ascii\_art, filename):

            Save the generated ASCII art to a file.

    """

    def \_\_init\_\_(self, letters):

        """

        Initialize the ASCIIArtGenerator object.

        Args:

            letters (dict): Dictionary containing ASCII representations of letters.

        Returns:

            None

        Raises:

            None

        """

        self.letters = letters

    @staticmethod

    def generate\_colored\_ascii\_art(text, letters, width, height, alignment="left"):

        """

        Generate colored ASCII art from the given text.

        Args:

            text (str): The text to be converted into colored ASCII art.

            letters (dict): Dictionary containing ASCII representations of letters.

            width (int): Width of the generated ASCII art.

            height (int): Height of the generated ASCII art.

            alignment (str, optional): Alignment of the ASCII art (default is "left").

        Returns:

            str: Colored ASCII art string.

        Raises:

            None

        """

        colors = {

            "R": "\x1b[31m",  # Red

            "G": "\x1b[32m",  # Green

            "Y": "\x1b[33m",  # Yellow

            "B": "\x1b[34m",  # Blue

            "M": "\x1b[35m",  # Magenta

            "C": "\x1b[36m",  # Cyan

            "W": "\x1b[37m"   # White

        }

        text = text.upper()

        ascii\_art = []

        real\_letter\_width = len(letters["A"][0])

        if width < len(text) \* real\_letter\_width:

            real\_letter\_width = width // len(text)

        for line in range(5):

            art\_line = ""

            for char in text:

                if char in letters:

                    letter = letters[char]

                    color\_code = (ord(char) % len(colors))

                    colored\_char = f"{colors[list(colors.keys())[color\_code]]}{letter[line][:real\_letter\_width]}\x1b[0m"

                    art\_line += colored\_char

                else:

                    art\_line += " " \* real\_letter\_width

            if alignment == "left":

                ascii\_art.append(art\_line)

            elif alignment == "center":

                left\_padding = (width - len(art\_line)) // 2

                right\_padding = width - len(art\_line) - left\_padding

                centered\_line = " " \* left\_padding + art\_line + " " \* right\_padding

                ascii\_art.append(centered\_line)

            elif alignment == "right":

                right\_padding = width - len(art\_line)

                right\_aligned\_line = " " \* right\_padding + art\_line

                ascii\_art.append(right\_aligned\_line)

        scaled\_ascii\_art = []

        for line in ascii\_art:

            for \_ in range(height):

                scaled\_ascii\_art.append(line)

        return "\n".join(scaled\_ascii\_art)

    @staticmethod

    def save\_ascii\_art\_to\_file(ascii\_art, filename):

        """

        Save the generated ASCII art to a file.

        Args:

            ascii\_art (str): The ASCII art to be saved.

            filename (str): The name of the file to save the ASCII art to.

        Returns:

            None

        Raises:

            IOError: If an error occurs during the file saving process.

        """

        try:

            with open(filename, 'w') as file:

                file.write(ascii\_art)

            print(f"ASCII art saved to the file {filename}")

        except IOError as e:

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

**main.py**

from .user\_input import UserInput

from .ascii\_art\_generator import ASCIIArtGenerator

from tkinter import filedialog, Tk

def main():

    """

    Main function to interact with the user, generate, and optionally save ASCII art.

    This function prompts the user for input, including the word or phrase to convert to ASCII art,

    the size of the ASCII art, and the alignment preference. It then generates colored ASCII art

    and offers the option to save the art to a file.

    Raises:

        None

    """

    letters = {

        "A": ["   A   ", "  A A  ", " A   A ", " AAAAA ", " A   A за"],

        "B": ["  BBB  ", " B   B ", "  BBB  ", " B   B ", "  BBB  "],

        "C": ["  CCC  ", "  C    ", " C    ", "  C    ", "   CCC  "],

        "D": [" DD   ", " D  D  ", " D   D ", " D  D  ", " DD   "],

        "E": [" EEEE ", " E    ", " EEE  ", " E    ", " EEEE "],

        "F": [" FFFF ", " F    ", " FFF  ", " F    ", " F    "],

        "G": ["  GGG ", " G    ", " G  GG", " G   G", "  GGGG"],

        "H": [" H   H ", " H   H ", " HHHHH ", " H   H ", " H   H "],

        "I": [" III  ", "  I   ", "  I   ", "  I   ", " III  "],

        "J": ["  JJJ  ", "   J  ", "   J  ", " J J  ", "  JJ  "],

        "K": [" K  K ", " K K  ", " KK   ", " K K  ", " K  K "],

        "L": [" L    ", " L    ", " L    ", " L    ", " LLLL "],

        "M": [" M   M ", " MM MM ", " M M M ", " M   M ", " M   M "],

        "N": [" N   N ", " NN  N ", " N N N ", " N  NN ", " N   N "],

        "O": ["  O  ", " O O ", " O O ", " O O ", "  O  "],

        "P": [" PPP  ", " P   P ", " PPP  ", " P    ", " P    "],

        "Q": ["  QQ  ", " Q  Q ", " Q  Q ", " Q QQ ", "  QQ Q"],

        "R": [" RRR  ", " R  R ", " RRR  ", " R R  ", " R  R "],

        "S": ["  SSS  ", " S    ", "  SSS ", "     S", "  SSS "],

        "T": [" TTTTT ", "   T  ", "   T  ", "   T  ", "   T  "],

        "U": [" U   U ", " U   U ", " U   U ", " U   U ", "  UUU "],

        "V": [" V   V ", " V   V ", " V   V ", "  V V  ", "   V  "],

        "W": [" W   W ", " W   W ", " W W W ", " W W W ", "  W W "],

        "X": [" X   X ", " X   X ", "  X X  ", " X   X ", " X   X "],

        "Y": [" Y   Y ", " Y   Y ", "  YYY  ", "   Y  ", "   Y  "],

        "Z": [" ZZZZ  ", "   Z  ", "  Z   ", " Z    ", " ZZZZ "],

        "А": ["   А   ", "  А А  ", " А   А ", " ААААА ", " А   А "],

        "Б": ["  БББ  ", " Б   Б ", "  БББ  ", " Б   Б ", "  БББ  "],

        "В": [" ВВВ   ", " В   В ", " ВВВ   ", " В   В ", " ВВВ   "],

        "Г": [" ГГГГГ ", " Г     ", " Г     ", " Г     ", " Г     "],

        "Д": [" Д  Д  ", " Д  Д  ", " ДДДДД ", " Д   Д ", " Д   Д "],

        "Е": [" ЕЕЕЕ ", " Е    ", " ЕЕЕ  ", " Е    ", " ЕЕЕЕ "],

        "Є": ["  ЄЄЄЄЄ ", " Є      ", " ЄЄЄЄ   ", " Є      ", "  ЄЄЄЄЄ "],

        "Ж": [" Ж   Ж   Ж ", "  Ж Ж    Ж  ", "   ЖЖЖЖЖ   ", "  Ж Ж    Ж  ", " Ж   Ж   Ж "],

        "З": [" ЗЗЗ ", "   З  ", "  З   ", "   З  ", " ЗЗЗ "],

        "И": [" И   И ", " И  I  ", " И I I ", " И   I ", " И   I "],

        "І": [" ІІІ ", "  І  ", "  І  ", "  І  ", " ІІІ "],

        "Ї": ["   Ї   ", "   Ї   ", "   Ї   ", "  ЇЇЇ  ", "  ЇЇЇ  "],

        "Й": [" Й   Й ", " Й   Й ", " ЙЙЙЙЙ ", " Й   Й ", " Й   Й "],

        "К": [" К   К ", " К  К  ", " ККК  ", " К  К  ", " К   К "],

        "Л": [" Л   Л ", " Л   Л ", " Л   Л ", " Л   Л ", " ЛЛЛЛ "],

        "М": [" М   М   М ", " ММ ММ  ММ ", " М М М М М ", " М  M  M  М ", " М   M   М "],

        "Н": [" Н   Н ", " Н   Н ", " ННННН ", " Н   Н ", " Н   Н "],

        "О": ["  ООО  ", " О   О ", " О   О ", " О   О ", "   ООО  "],

        "П": ["  ППП  ", " П   П ", " П   П ", " П   П ", " П   П"],

        "Р": [" РРР  ", " Р   Р ", " РРР  ", " Р     ", " Р     "],

        "С": ["  ССС  ", " С     ", " С     ", " С     ", "  ССС  "],

        "Т": [" ТТТТТ ", "   Т   ", "   Т   ", "   Т   ", "   Т   "],

        "У": [" У   У ", " У   У ", " У   У ", " У У У ", "  УУУ  "],

        "Ф": ["   ФФФ   ", "  Ф   Ф  ", " ФФФФФФ ", " Ф   Ф  ", " Ф   Ф  "],

        "Х": [" Х   Х ", "  ХХХ  ", "   Х   ", "  ХХХ  ", " Х   Х "],

        "Ц": [" Ц   Ц   Ц ", "  Ц   Ц   Ц  ", "  Ц   Ц   Ц  ", " ЦЦЦЦЦЦ ", "       Ц  "],

        "Ч": [" Ч   Ч ", "  Ч   Ч ", "  Ч   Ч ", "   ЧЧЧ  ", "   ЧЧ   "],

        "Ш": [" Ш     Ш ", "  Ш   Ш  ", "   Ш Ш   ", "    Ш    ", " Ш     Ш "],

        "Щ": [" Щ     Щ     Щ ", "  Щ   Щ   Щ  ", "   Щ ЩЩ Щ   ", "    Щ   Щ    ", " Щ     Щ     Щ "],

        "Ь": [" Ь   Ь ", " Ь   Ь ", " Ь   Ь ", " Ь ЬЬ  ", "  Ь   Ь "],

        "Ю": [" Ю   Ю ", " Ю   Ю ", "  ЮЮЮ  ", " Ю   Ю ", " Ю   Ю "],

        "Я": ["  ЯЯЯ ", " Я   Я ", "  ЯЯЯ ", "     Я", " ЯЯЯ "],

    }

    user\_input = input("Enter a word or phrase to convert to ASCII art: ")

    width, height = UserInput.get\_valid\_size()

    alignment\_choice = UserInput.get\_alignment\_choice()

    if alignment\_choice == "1":

        alignment = "left"

    elif alignment\_choice == "2":

        alignment = "center"

    elif alignment\_choice == "3":

        alignment = "right"

    ascii\_art = ASCIIArtGenerator.generate\_colored\_ascii\_art(user\_input, letters, width, height, alignment)

    print(ascii\_art)

    save\_to\_file = input("Save ASCII art to a file? (Yes/No): ").strip().lower()

    if save\_to\_file == "yes":

        file\_path = filedialog.asksaveasfilename(

            defaultextension=".txt",

            filetypes=[("Text files", "\*.txt")],

            initialdir="data/lab4",

            initialfile="art"

        )

        if file\_path:

            ASCIIArtGenerator.save\_ascii\_art\_to\_file(ascii\_art, file\_path)  # Save the ASCII art to the specified file

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**user\_input.py**

class UserInput:

    @staticmethod

    def get\_valid\_size():

        """

        Get valid width and height for ASCII art from user input.

        Returns:

            Tuple[int, int]: A tuple containing width and height.

        Raises:

            ValueError: If input values are not valid integers or are non-positive.

        """

        while True:

            try:

                width = int(input("Enter the width of the ASCII art: "))

                height = int(input("Enter the height of the ASCII art: "))

                if width > 0 and height > 0:

                    return width, height

                else:

                    print("Dimensions should be greater than 0.")

            except ValueError:

                print("Please enter valid integers for dimensions.")

    @staticmethod

    def get\_alignment\_choice():

        """

        Get the user's choice for text alignment.

        Returns:

            str: The user's choice for alignment (either '1', '2', or '3').

        """

        while True:

            print("Choose alignment option:")

            print("1. Left alignment")

            print("2. Center alignment")

            print("3. Right alignment")

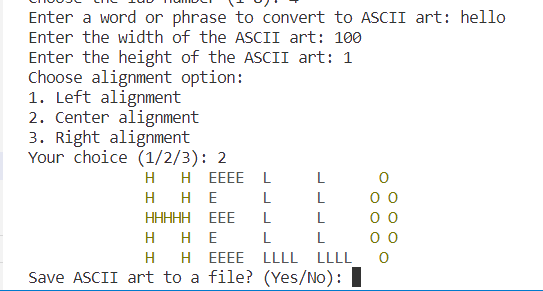
            choice = input("Your choice (1/2/3): ")

            if choice in ["1", "2", "3"]:

                return choice

            else:

                print("Please choose one of the options (1/2/3).")



*Рис. 1 – Результат створення арту*

**Висновок:** на даній лабораторній роботі я створила генератор ASCII-арту без використання зовнішніх бібліотек.