МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра інформаційних систем та мереж

Лабораторна робота №3

з дисципліни: «Спеціалізовані мови програмування»

на тему: «Розробка ASCII ART генератора для візуалізації текстових даних »

Виконала:
студентка групи ІТ-32
Моляща Ю.А.
Прийняв:
Щербак С.С.

Львів - 2023

Лабораторна робота №3

«Розробка ASCII ART генератора для візуалізації текстових даних »

Мета роботи: створення додатка Генератора ASCII-арту.

Завдання на лабораторну роботу.

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.

Хід роботи.

Код програми:

ascii_art_generator.py

```
"""ASCII-art generator."""


import os

import pyfiglet

from pyfiglet import Figlet

from colorama import Fore


# To initialize colorama and configure its behavior

from colorama import init as colorama_init


# Initialize colorama

# autoreset=True -> Color settings will automatically reset after each print
statement

colorama_init(autoreset=True)


# FOLDER_PATH = 'source/lab3/ASCII-arts/'


def settings(settings_obj):

    while True:

        print('Options (1/2/3/4/5/6):')

        print('0. Show current settings')

        print('1. Change font')

        print('2. Change size')

        print('3. Change symbol')

        print('4. Change color')

        print('5. Reset settings')

        print('6. Back')


        user_input = input('Enter option number: ')
```

```
        if user_input == '0':
            settings_obj.show_settings()
        if user_input == '1':
            settings_obj.set_font(set_font())
        elif user_input == '2':
            settings_obj.set_size(*set_size())
        elif user_input == '3':
            settings_obj.set_symbols(*set_symbols())
        elif user_input == '4':
            settings_obj.set_color(set_color())
        elif user_input == '5':
            settings_obj.default_settings()
        elif user_input == '6':
            break

def get_phrase():
    """Get user input.

    Returns:
        user_input (str): User input.
    """
    user_input = input('Enter the phrase: ')

    return user_input

def create_ascii_art(FOLDER_PATH, settings_obj):
    """Ask user for phrase and show ASCII-art.

    Args:
        settings_obj (AsciiArtSettings): Object with settings.
    """
    user_input = get_phrase()
    width, height = settings_obj.size
    try:
```

```
        check_size(user_input, width, height)
    except ValueError as e:
        print(f"An error occurred: {e}")
        return None

    art = Figlet(font=settings_obj.font, width=settings_obj.size[0])
    art = art.renderText(user_input)

    if settings_obj.symbols:
        art = change_symbols(art, settings_obj.symbols[0])

    art = settings_obj.color + art

    preview_art(FOLDER_PATH, art)

def set_font():
    """Set font for ASCII-art.

    Returns:
        font (str): Font name.
    """
    # Get list of font names
    fonts = pyfiglet.FigletFont.getFonts()

    # Enumerate and print fonts with numbering, starting from 1
    for index, font in enumerate(fonts, start=1):
        print(f"{index}. {font}")

    user_input = int(input('Enter the font number: '))
    font = fonts[user_input-1]

    return font

def set_size():
```

```
    """Set size for ASCII-art.

    Returns:
        width (int): Width of ASCII-art.
        height (int): Height of ASCII-art.
    """
    width = int(input('Enter width: '))
    height = int(input('Enter height: '))

    return width, height

def check_size(char_str, width, height):
    char_width = 8
    char_height = 8

    str_length = len(char_str) * char_width
    terminal_columns, terminal_lines = os.get_terminal_size()

    if width < char_width:
        raise ValueError(f"Width {width} is too small for the string length of
{str_length}")
    elif width > terminal_columns:
        raise ValueError(f"Width {width} exceeds the terminal length of
{terminal_columns}")
    else:
        pass

    if height < char_height or height < str_length / width:
        raise ValueError(f"Height {height} is too small for the string length
of {str_length}")
    elif height > terminal_lines:
        raise ValueError(f"Height {height} exceeds the terminal length of
{str_length}")
    else:
        pass
```

```
def set_symbols():
    """Set symbol for ASCII-art.

    Returns:
        symbol (str): Symbol to replace.
    """
    regular_symbol = input('Enter regular symbol: ')

    set_shadow = input('Do you want to set a shadow symbol? (y/n): ')
    if set_shadow.lower() == 'y':
        shadow_symbol = input('Enter shadow symbol: ')
    else:
        shadow_symbol = ''

    # Get list of ASCII symbols
    ascii_dec_values = [i for i in range(0, 256)]
    ascii_symbols = [chr(i) for i in ascii_dec_values]

    # Check if symbol is in ASCII
    if regular_symbol and shadow_symbol not in ascii_symbols and shadow_symbol
!= '':
        print('Symbol is not in ASCII.')
        return None
    else:
        print("Symbol changed!")
        return regular_symbol, shadow_symbol


def set_alignment():
    """Set alignment for ASCII-art.

    Returns:
        alignment (str): Alignment.
    """
```

```
    alignment = input('Enter alignment (left/center/right): ')

    return alignment

def set_3d_option():
    while True:
        user_input = input('3D option (y/n): ')
        if user_input == 'y':
            return True
        elif user_input == 'n':
            return False
        else:
            print('Invalid input. Please enter y or n.')

def change_symbols(art, symbol):
    """Change symbols in ASCII-art.

    Args:
        art (str): ASCII-art.
        symbol (str): Symbol to replace.

    Returns:
        art (str): ASCII-art with replaced symbols.
    """
    for char in art:
        if char != '\n' and char != ' ':
            art = art.replace(char, symbol)

    return art

def set_color():
    """Set color for ASCII-art.

    Returns:
```

```
        color_code (str): Color code.

    Raises:
        IndexError: If color number is not in range.
    """
    # Get dictinary where key is color name and value is color code
    colors = dict(Fore.__dict__.items())

    # Enumerate and print colors with numbering, starting from 1
    for index, color in enumerate(colors.keys(), start=1):
        print(f"{index}. {colors[color]}{color}")

    user_input = int(input('Enter the color number: '))

    # Get user input
    try:
        color_code = colors[list(colors.keys())[user_input-1]]
        return color_code
    except IndexError:
        print(f'Color number is not in range. Available colors are in range
from 1 to {len(colors)}.')

def set_alignment():
    """Set alignment for ASCII-art.

    Returns:
        alignment (str): Alignment.
    """
    alignment = input('Enter alignment (left/center/right): ')

    return alignment


def preview_art(FOLDER_PATH, art):
```

```
    """Preview ASCII-art.

    Args:
        art (str): ASCII-art.
    """
    print(art)

    save_art_answ = input('Do you want to save your art? (y/n): ')

    if save_art_answ == 'y':
        save_art(FOLDER_PATH, art)
    else:
        pass

def save_art(FOLDER_PATH, art):
    """Save ASCII-art to file.

    Args:
        art (str): ASCII-art.
    """
    file_name = input('Give a file name: ')
    formated_file_name = FOLDER_PATH + file_name + '.txt'

    with open(formated_file_name, 'w') as file:
        file.write(art)

def show_art(FOLDER_PATH):
    """Show ASCII-art from file.

    Raises:
        FileNotFoundError: If file not found.
    """
    try:
        file_name = input('Enter file name: ')
```

```
        formated_file_name = FOLDER_PATH + file_name + '.txt'
        with open(formated_file_name, 'r') as file:
            print(file.read())
    except FileNotFoundError:
        print('File not found.')
```

ascii_art_settings.py

```
"""Module for storing settings for ASCII-art."""
import json

class AsciiArtSettings:
    """Class for storing settings for ASCII-art.

    Attributes:
        font (str): Font name.
        size (tuple): Size of ASCII-art.
        symbols (str): Symbols to replace.
        color (str): Color of ASCII-art.
    """
    def __init__(self):
        self.settings_file_path = None
        self.font = None
        self.size = None
        self.symbols = None
        self.color = None
        self.alignment = None
        self.is_3d = None

    def set_settings_file_path(self, settings_file_path):
        self.settings_file_path = settings_file_path

    def show_settings(self):
        print('Current settings:')
        print('Font:', self.font)
        print('Size:', self.size)
```

```
        print('Symbols:', self.symbols)
        print('Alignment:', self.alignment)
        print('Color:', self.color.replace('\x1b', '\\x1b'))
        print('3D:', self.is_3d)

    def set_font(self, font):
        self.font = font
        self.save_settings()

    def set_size(self, width, height):
        self.size = (width, height)
        self.save_settings()

    def set_symbols(self, regular_symbol, shadow_symbol):
        self.symbols = (regular_symbol, shadow_symbol)
        self.save_settings()

    def set_color(self, color):
        self.color = color
        self.save_settings()

    def set_alignment(self, alignment):
        self.alignment = alignment
        self.save_settings()

    def set_3d_option(self, is_3d):
        self.is_3d = is_3d
        self.save_settings()

    def default_settings(self):
        self.font = 'clb6x10'
        self.size = (80, 25)
        self.symbols = ("#", "*")
        self.color = '\x1b[39m'
```

```
        self.alignment = 'left'
        self.is_3d = False
        self.save_settings()

    def save_settings(self):
        settings_data = {
            'font': self.font,
            'size': self.size,
            'symbols': self.symbols,
            'color': self.color,
            'alignment': self.alignment,
            'is_3d': self.is_3d
        }
        with open(self.settings_file_path, 'w') as file:
            json.dump(settings_data, file)

    def load_settings(self):
        try:
            with open(self.settings_file_path, 'r') as file:
                settings_data = json.load(file)
                self.font = settings_data['font']
                self.size = settings_data['size']
                self.symbols = settings_data['symbols']
                self.color = settings_data['color']
                self.alignment = settings_data['alignment']
                self.is_3d = settings_data['is_3d']
        except FileNotFoundError:
            # If file does not exist use default settings
            self.default_settings()
```

main.py

```
""" Main module for the ASCII-art program."""

# Include the parent directory in the system's import path
import sys
```

```
import os

current_dir = os.path.dirname(os.path.abspath(__file__))
parent_dir = os.path.abspath(os.path.join(current_dir, '..'))
sys.path.append(parent_dir)

from lab3.ascii_art_settings import AsciiArtSettings
from lab3.ascii_art_generator import create_ascii_art, show_art, settings

FOLDER_PATH = 'source/lab3/ASCII-arts/'
SETTINGS_FILE_PATH = 'source/lab3/settings.json'

def main():
    settings_obj = AsciiArtSettings()
    settings_obj.set_settings_file_path(SETTINGS_FILE_PATH)
    settings_obj.load_settings()

    while True:
        print('Options (1/2/3):')
        print('1. Create ASCII-art')
        print('2. Show ASCII-art')
        print('3. Settings')
        print('4. Exit')

        user_input = input('Enter option number: ')

        if user_input == '1':
            create_ascii_art(FOLDER_PATH, settings_obj)
        elif user_input == '2':
            show_art(FOLDER_PATH)
        elif user_input == '3':
            settings(settings_obj, SETTINGS_FILE_PATH)
        elif user_input == '4':
            break
```

```
if __name__ == "__main__":
    main()
```

settings.json

```
{"font": "clb6x10", "size": [80, 25], "symbols": ["#", "*"], "color":
"\u001b[39m", "alignment": "left", "is_3d": false}
```

Приклад виконання програми зображено на рис. 1.

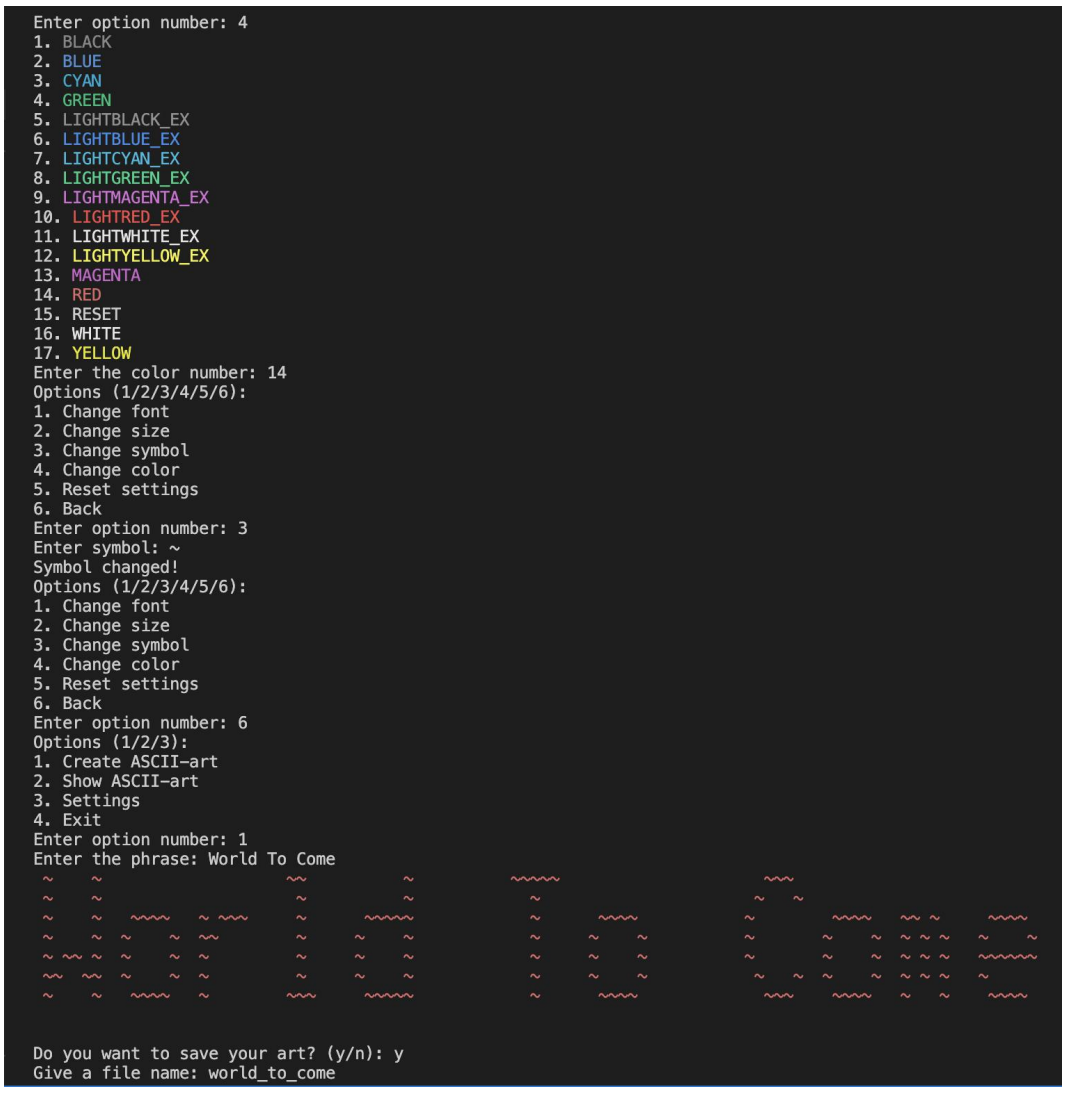

Рис. 1. Приклад виконання програми

Висновок: під час виконання лабораторної роботи було створено додаток генератор ASCII-арту.