Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Лабораторна роботи №4

з дисципліни «Спеціалізовані мови програмування»

на тему

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

Виконав:

Костик В. Ю.

Перевірив:

Щербак С. С.

Львів 2024

**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.

**Реалізація:**

**Папка AppSettings, файл AppSettings.py:**  
DEFAULT\_WIDTH = 80

DEFAULT\_SYMBOL = '#'

COLORS = {

1: 'Червоний',

2: 'Синій',

3: 'Зелений',

4: 'Білий',

5: 'Жовтий'

}

**Папка classes, файл ASCIIArt.py:**

class ASCIIArt:

def \_\_init\_\_(self, text, font="basic", color="Білий", width=80, custom\_char="#"):

self.text = text

self.font = font

self.color = color

self.width = width

self.custom\_char = custom\_char

self.ascii\_art = ""

self.color\_codes = {

"Червоний": "\033[91m",

"Синій": "\033[94m",

"Зелений": "\033[92m",

"Білий": "\033[97m",

"Жовтий": "\033[93m",

}

self.reset\_code = "\033[0m"

def generate\_art(self):

"""Генерує ASCII-арт за допомогою вибраного шрифту, вирівнюючи символи в одному рядку."""

lines = [""] \* 5 # Кількість рядків у символах шрифту (в даному випадку 5)

# Перетворюємо кожен символ на його ASCII представлення

for char in self.text:

ascii\_char = self.\_convert\_char\_to\_ascii(char)

ascii\_lines = ascii\_char.split("\n")

# Додаємо кожен рядок ASCII символу до відповідного рядка в загальному арті

for i in range(len(ascii\_lines)):

lines[i] += ascii\_lines[i]

self.ascii\_art = "\n".join(lines)

self.\_apply\_custom\_characters()

return self.\_apply\_color(self.ascii\_art)

def \_convert\_char\_to\_ascii(self, char):

"""Перетворює символ у його ASCII подання на основі вибраного шрифту."""

fonts = {

'basic': {

'A': [

" # ",

" # # ",

" # # ",

" ##### ",

" # # "

],

#решта букв і цифр

'block': {

'A': [

" ███ ",

" █ █ ",

" █████ ",

" █ █ ",

" █ █ "

#решта букв і цифр

],

},

'slant': {

'A': [

" /\ ",

" / \ ",

" /\_\_\_\_\ ",

" / \ ",

" / \ "

],

#решта букв і цифр

}

}

font\_dict = fonts.get(self.font, fonts['basic']) # За замовчуванням шрифт basic

return "\n".join(font\_dict.get(char.upper(), [char]))

def \_apply\_custom\_characters(self):

"""Замінює стандартні символи на користувацькі."""

self.ascii\_art = self.ascii\_art.replace('#', str(self.custom\_char)).replace('█', str(self.custom\_char))

#.replace('|', str(self.custom\_char)).replace('-', str(self.custom\_char)).replace('|', str(self.custom\_char)).replace('/', str(self.custom\_char)).replace('\_', str(self.custom\_char)).replace('\\', str(self.custom\_char))

def \_apply\_color(self, art):

"""Застосовує вибраний колір до ASCII-арту."""

color\_code = self.color\_codes.get(self.color, self.color\_codes["Білий"])

return f"{color\_code}{art}{self.reset\_code}"

def preview(self):

print("Попередній перегляд ASCII-арту:")

print(self.generate\_art())

def save\_to\_file(self, file\_path):

with open(file\_path, "w", encoding="utf-8") as f:

f.write(self.ascii\_art)

print(f"ASCII-арт збережено у файл {file\_path}")

**Папка classes, файл ASCIIArtApp.py:**

from classes.ASCIIArt import ASCIIArt

from functions import functions

class ASCIIArtApp:

def \_\_init\_\_(self):

self.colors = {

1: 'Червоний',

2: 'Синій',

3: 'Зелений',

4: 'Білий',

5: 'Жовтий'

}

def get\_font\_choice(self):

fonts = functions.get\_available\_fonts()

return functions.font\_selection(fonts)

def get\_user\_input(self):

text = functions.get\_validated\_text()

print("Доступні шрифти:")

selected\_font = self.get\_font\_choice()

custom\_char = functions.get\_validated\_custom\_char()

width = functions.get\_width()

# Вибір кольору

color = functions.get\_color\_choice(self.colors)

# Повертаємо об'єкт ASCIIArt

return ASCIIArt(text, selected\_font, color, width, custom\_char)

def run(self):

# Виклик методу для отримання введених даних

art = self.get\_user\_input()

# Генерація та відображення арту

print(art.generate\_art())

save\_choice = functions.get\_save\_choice()

if save\_choice == "так":

file\_path = r"C:\\Users\\Vlad\\Desktop\\унік\\пітоній\\ASIIART\_Pro\_Max\_beta\_v04\\art.txt"

art.save\_to\_file(file\_path)

art.preview()

**Папка functions, файл functions.py:**from AppSettings.AppSettings import COLORS

def get\_available\_fonts():

"""Повертає список доступних шрифтів для ASCII-арту."""

return ["basic", "block", "slant"]

def preview\_font\_example(font):

"""Показує приклад шрифту."""

examples = {

"basic": "Example:\n ### \n # # \n ##### \n # # \n # # \n",

"block": "Example:\n ███ \n █ █ \n █████ \n █ █ \n █ █ \n",

"slant": "Example:\n /\\ \n / \\ \n /\_\_\_\_\\ \n / \\ \n/ \\\n"

}

return examples.get(font, "Приклад не знайдено.")

def validate\_latin\_input(text):

"""Перевірка, чи містить введений текст тільки латинські символи."""

for char in text:

# Перевіряємо, чи символ є латинським символом або цифрою

if not (('a' <= char <= 'z') or ('A' <= char <= 'Z') or ('0' <= char <= '9') or char.isspace()):

print("Помилка: Текст має містити тільки латинські символи та цифри.")

return False

return True

def font\_selection(fonts):

print("Доступні шрифти:")

for i, font in enumerate(fonts):

print(f"{i + 1}. {font}")

while True:

choice = input("Виберіть шрифт: ")

try:

selected\_font = fonts[int(choice) - 1]

print(f"Приклад шрифту:\n{preview\_font\_example(selected\_font)}")

return selected\_font

except (IndexError, ValueError):

print("Помилка: Виберіть правильний номер шрифту.")

def get\_validated\_text():

while True:

text = input("Введіть слово або фразу для генерації ASCII-арту (латиницею): ")

if validate\_latin\_input(text):

return text

def validate\_custom\_char(char):

"""Перевірка, чи введений символ не містить кирилицю."""

# Тут ми можемо перевірити, чи символ не є кириличним.

if 'а' <= char <= 'я' or 'А' <= char <= 'Я':

print("Помилка: Символ не має містити кириличні символи.")

return False

return True

def get\_validated\_custom\_char():

"""Отримує і перевіряє символ для ASCII-арту."""

while True:

custom\_char = input("Введіть символ для ASCII-арту, або натисніть Enter для використання # (працює тільки з шрифтами basic та block): ")

if not custom\_char: # Якщо нічого не введено, використати '#'

return '#'

if len(custom\_char) != 1: # Перевіряємо, що введено лише один символ

print("Помилка: Ви повинні ввести рівно один символ.")

continue

if validate\_custom\_char(custom\_char): # Перевіряємо введений символ

return custom\_char

def get\_width():

"""Отримує ширину для ASCII-арту, або повертає значення за замовчуванням."""

default\_width = 80 # Значення за замовчуванням

while True:

width\_input = input(f"Введіть ширину ASCII-арту (за замовчуванням {default\_width}): ")

if not width\_input: # Якщо нічого не введено, використати значення за замовчуванням

return default\_width

try:

width = int(width\_input) # Спробуйте перетворити на число

if width > 0: # Переконайтесь, що ширина більше нуля

return width

else:

print("Помилка: Введіть число більше нуля.")

except ValueError:

print("Помилка: Введіть число для ширини.")

def get\_color\_choice(colors):

while True:

print("Виберіть колір:")

for num, color in colors.items():

print(f"{num}: {color}")

try:

color\_choice = int(input("Введіть номер кольору: "))

if color\_choice in colors:

return colors[color\_choice]

else:

print("Помилка: Виберіть номер кольору з наявних.")

except ValueError:

print("Помилка: Введіть число.")

def get\_save\_choice():

"""Запитує у користувача, чи хоче він зберегти ASCII-арт у файл."""

while True:

save\_choice = input("Бажаєте зберегти ASCII-арт у файл? (так/ні): ").strip().lower()

if save\_choice in ["так", "ні"]:

return save\_choice

print("Помилка: Введіть 'так' або 'ні'.")

**Папка interface, файл interface.py:**

def display\_menu():

print("Вітаємо в Генераторі ASCII-арту!")

print("Оберіть необхідну дію:")

**Папка logs, файл logs.py:**

def log\_action(action):

with open("log.txt", "a") as log\_file:

log\_file.write(action + "\n")

**Файл main.py:**

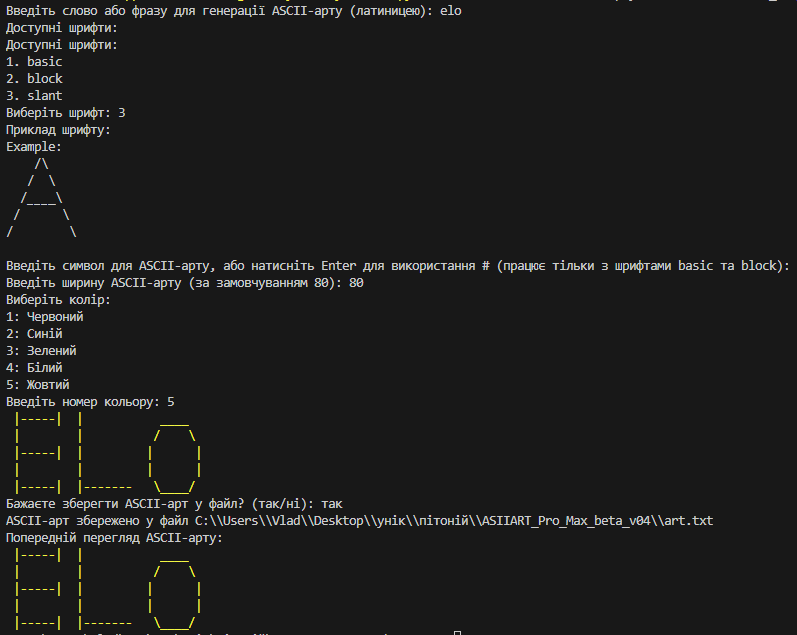
from classes.ASCIIArtApp import ASCIIArtApp

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

app = ASCIIArtApp()

app.run()

**Результат виконання:**

****

**Висновок:** на цій лабораторній роботі я створив генератор ASCII-арту з нуля, та надав можливість налаштовувати символи, розміри, вирівнювання та кольори