Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ІСМ



Звіт

про виконання лабораторної роботи № 4

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

з дисципліни

«Спеціалізовані мови програмування»

Виконав:

Студент групи ІТ-32,

Садовенко А. Ю.

Прийняв:

Щербак С.С

Львів 2023

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

**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.

**Код:**

**main.py:**

from ascii\_art\_generator import \*

def save\_to\_file(destination\_path, content) -> None:

with open(destination\_path, "w") as output\_file:

output\_file.write(content)

def load\_from\_file(source\_path) -> str:

with open(source\_path, "r") as input\_file:

return input\_file.read()

def main():

try:

user\_input\_text = str(input("Input text to convert to ASCII art: "))

AbstractDataHandler.show\_color\_options()

chosen\_color\_index = int(input("Choose a color index: "))

desired\_width = int(input("Set the width for ASCII art: "))

alphabet\_file\_path = "ascii\_alphabet.txt"

ascii\_processor = TextFileHandler(alphabet\_file\_path)

ascii\_art = ascii\_processor.fetch(user\_input\_text, chosen\_color\_index, desired\_width)

print(ascii\_art)

save\_to\_file("ascii\_output.txt", ascii\_art)

except KeyError:

print("Error: Incorrect color index or missing characters in the ASCII file.")

except ValueError as ve:

print(ve)

except FileNotFoundError:

print("Error: ASCII character file not found. Please verify the path.")

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

main()  
  
**ascii\_art\_generator.py:**

import os

import re

from abc import ABC, abstractmethod

import colorama

from colorama import Fore

from functools import reduce

colorama.init(autoreset=True)

color\_palette = {index: color for index, color in enumerate(sorted(Fore.\_\_dict\_\_.keys()))}

class AbstractDataHandler(ABC):

def \_\_init\_\_(self, path\_to\_file):

if not path\_to\_file:

raise ValueError("Path to file cannot be empty")

self.\_file\_path = path\_to\_file

@abstractmethod

def \_read\_and\_process\_data(self) -> None:

pass

@abstractmethod

def fetch(self, text, color\_index, line\_width) -> str:

pass

def \_load\_file\_content(self) -> str:

with open(self.\_file\_path, 'r') as file:

return file.read()

@staticmethod

def show\_color\_options() -> None:

for index, color\_name in color\_palette.items():

print(f"{index}. {color\_name}")

class TextFileHandler(AbstractDataHandler):

\_meta\_info = {}

\_content = {}

def \_\_init\_\_(self, path\_to\_file):

if not path\_to\_file.endswith(".txt") or not os.path.exists(path\_to\_file):

raise ValueError("File must be a .txt file and exist")

super().\_\_init\_\_(path\_to\_file)

self.\_read\_and\_process\_data()

def \_read\_and\_process\_data(self) -> None:

file\_content = self.\_load\_file\_content().split("\n")

data\_section\_found = False

symbol\_representation = ""

current\_symbol = None

for line in file\_content:

if data\_section\_found:

if re.match("^@symbol::.$", line):

if current\_symbol:

self.\_content[current\_symbol] = symbol\_representation

current\_symbol = line[-1]

symbol\_representation = ""

row\_length = 0

line\_counter = 1

elif re.match("^\\^.+\\$$", line):

if line\_counter == 1:

row\_length = len(line)

if len(line) != row\_length:

raise ValueError("Inconsistent row length for symbol representation")

symbol\_representation += line[1:-1] + ("\n" if line\_counter < self.\_meta\_info["height"] else "")

line\_counter += 1

else:

raise ValueError("Incorrect format in data section")

elif line == "@data":

if "height" not in self.\_meta\_info:

raise ValueError("Meta information 'height' is missing")

data\_section\_found = True

elif not data\_section\_found:

if re.match("^\\w+::\\d+$", line):

key, value = line.split("::")

self.\_meta\_info[key] = int(value)

else:

raise ValueError("Invalid meta data format")

def fetch(self, text, color\_index, max\_width) -> str:

symbol\_data = {}

needed\_symbols = {}

line\_properties = {}

current\_line = 0

current\_line\_width = 0

for index, char in enumerate(text):

symbol\_lines = self.\_content[char].split("\n")

if current\_line\_width + len(symbol\_lines[0]) > max\_width:

if len(symbol\_lines[0]) > max\_width:

raise ValueError("Text width too small")

current\_line += 1

current\_line\_width = 0

current\_line\_width += len(symbol\_lines[0])

line\_properties[current\_line] = line\_properties.get(current\_line, 0) + 1

needed\_symbols[index] = "\n".join(symbol\_lines)

symbol\_index = 0

for line in line\_properties:

for \_ in range(line\_properties[line]):

lines = needed\_symbols[symbol\_index].split("\n")

symbol\_index += 1

for row\_index, row in enumerate(lines):

line\_key = row\_index + line \* self.\_meta\_info["height"]

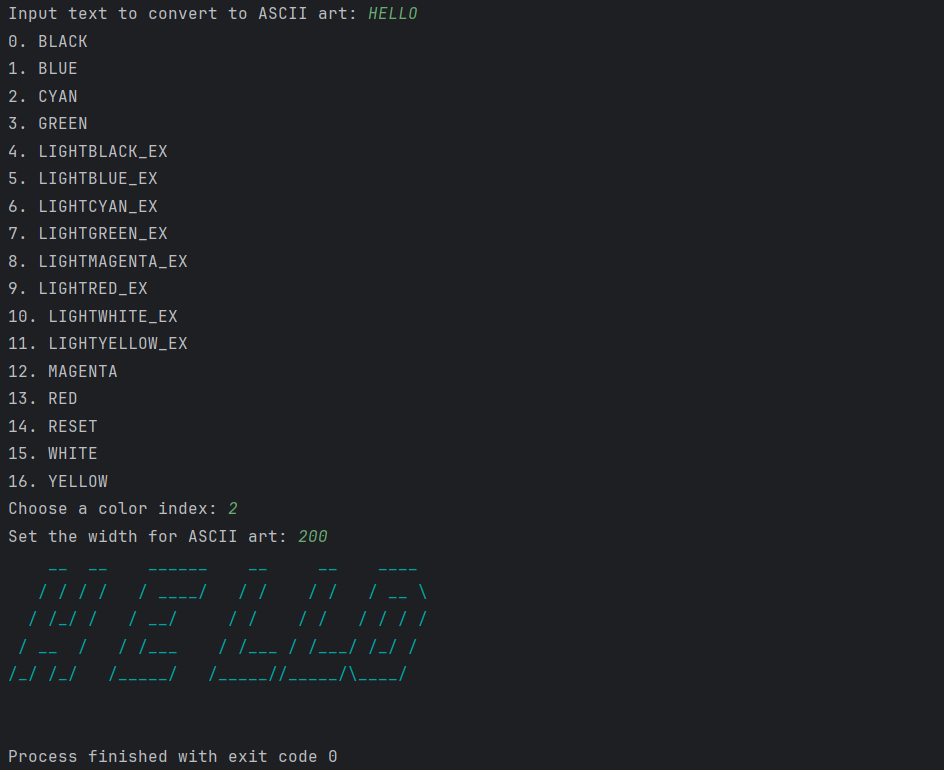
symbol\_data[line\_key] = symbol\_data.get(line\_key, "") + row

colored\_output = Fore.\_\_getattribute\_\_(color\_palette[color\_index])

return colored\_output + "\n".join(symbol\_data.values())

**Виконання програми**

На рис. 1 зображенно виконання програми:



*Рис. 1 Виконання програми*

**Посилання на GitHub репозиторій** [**https://github.com/ArtemSadovenko/smp/tree/master**](https://github.com/ArtemSadovenko/smp/tree/master)

**Висновки:** Виконуючи ці завдання, я створив генератор ASCII-арту з нуля, та створив можливість налаштовувати символи, розміри, вирівнювання та кольори, що дозволило мені глибше розібратися як створюється ASCII-арт.