Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 4

“ Розробка ASCII ART генератора для візуалізації 2D-фігур ”

з дисципліни **“ Спеціалізовані мови програмування”**

Виконала студентка групи ІТ-32
ДЕНИСЯК С.-М. Т.
Прийняв:
ЩЕРБАК С. С.

Львів – 2023

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 твореннями.

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 програми:

main.py:

```
from processors import *


def write_into_file(file_path, text) -> None:
    with open(file_path, "w") as file:
        file.write(text)


def read_from_file(file_path) -> str:
    with open(file_path, "r") as file:
        return file.read()


def main():
    try:
        initial_text = str(input("Enter text in order to display: "))
        DataProcessor.display_colors()
        color_position = int(input("Enter position of color you would like to
use: "))
        width = int(input("Enter width of text you would like to display: "))
        file_path = str(input("Enter path to file containing alphabet: "))
        txt_processor = TxtProcessor(file_path)
        text = txt_processor.retrieve(initial_text, color_position, width)
        print(text)
        write_into_file("output.txt", text)
    except KeyError:
        print("Key error! You have entered a wrong value for key of colors, "
              "or symbols are absent in the file which contains alphabet")
    except ValueError as e:
        print(e)
    except FileNotFoundError:
        print("File not found! Please, check the path to the file")


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

processors.py:

```
import os
from abc import ABC, abstractmethod
import colorama
from colorama import Fore
import re
from functools import reduce

colorama.init(autoreset=True)
colors = dict(enumerate(sorted(Fore.__dict__.keys())))


class DataProcessor(ABC):

    def __init__(self, file_path):
        if file_path is None:
            raise ValueError("File path should have a value")
        self.__file_path = file_path

    @abstractmethod
    def _get_all_data(self) -> None:
        pass

    @abstractmethod
    def retrieve(self, text, color, width) -> str:
        pass
```

```
    def _read_file(self) -> str:
        with open(self.__file_path, "r") as file:
            return file.read()

    @staticmethod
    def display_colors() -> None:
        for i in colors:
            print(str(i) + ". " + colors[i])


class TxtProcessor(DataProcessor):
    __meta_data = dict()
    __data = dict()

    """
    Raises:
        ValueError: If the file extension is incorrect, or the file_path is None
        FileNotFoundError: If the file does not exist in the file system
    """

    def __init__(self, file_path):
        if not file_path.endswith(".txt") and os.path.exists(file_path) is True:
            raise ValueError("File should be .txt file")
        super().__init__(file_path)
        self._get_all_data()

    """
    Retrieves all data from the file and stores it in the appropriate data
structure.

    Returns:
        None
    Raises:
        ValueError: If the file has incorrect format.
    """

    def _get_all_data(self) -> None:
        file_data = str(self._read_file()).split("\n")
        is_data_annotation_found = False
        representation = ""
        symbol = None

        for line in file_data:
            if is_data_annotation_found:
                if re.match("^@symbol:.$", line):
                    if symbol is not None:
                        self.__data[symbol] = representation
                    symbol = line[9:]
                    representation = ""
                    length = 0
                    counter = 1
                elif re.match("^\\^.+\\$$", line) is not None:
                    if counter == 1:
                        length = line.__sizeof__()
                    if line.__sizeof__() != length:
                        raise ValueError("Length of the row has to be equal
within a certain character in the file")
                    representation += line[1:-1] + ("" if counter ==
self.__meta_data["height"] else "\n")
                    counter += 1
                else:
                    raise ValueError("Data information has incorrect format")
            elif line == "@data":
                if not self.__meta_data.__contains__("height"):
                    raise ValueError("The file has to contain meta information
such as height")
```

```
                is_data_annotation_found = True
            elif not is_data_annotation_found:
                if re.match("^\\w+::\\d+$", line) is not None:
                    parts = line.split("::")
                    self.__meta_data[parts[0]] = int(parts[1])
                else:
                    raise ValueError("Metadata has incorrect format")

    """
    Retrieve the text representation of the given input text by formatting it
into rows with a maximum width.

    Args:
        text (str): The input text to be formatted.
        color_position (int): The index of the color in the `colors` list to be
applied to the formatted text.
        width (int): The maximum width of each row in the formatted text.

    Returns:
        str: The formatted text representation of the input text, with the
specified color applied.

    Raises:
        ValueError: If the width of the text is too small to fit within the
specified width.
        KeyError: If the color position is not a valid index in the `colors`
list, or if the symbol is absent in the file
    """

    def retrieve(self, text, color_position, width) -> str:
        result = dict()
        all_needed_symbols = dict()
        properties = dict()
        row_count = 0
        current_position_in_row = 0

        for i in range(0, len(text)):
            representation = str(self.__data[text[i]]).split("\n")
            if current_position_in_row + representation[0].__len__() > width:
                if representation[0].__len__() > width:
                    raise ValueError("Width of the text is too small")
                row_count += 1
                current_position_in_row = 0
                current_position_in_row += representation[0].__len__()
                if properties.__contains__(row_count):
                    properties[row_count] += 1
                else:
                    properties[row_count] = 1
            else:
                current_position_in_row += representation[0].__len__()
                if properties.__contains__(row_count):
                    properties[row_count] += 1
                else:
                    properties[row_count] = 1
            all_needed_symbols.update({i: reduce((lambda x, y: x + "\n" + y),
representation)})

        symbol_count = 0
        for i in properties.keys():
            for j in range(0, properties[i]):
                representation = all_needed_symbols[symbol_count].split("\n")
                symbol_count += 1
                for k in range(0, len(representation)):
                    if result.__contains__(k + i * 6):
                        result[k + i * 6] += representation[k]
                    else:
```

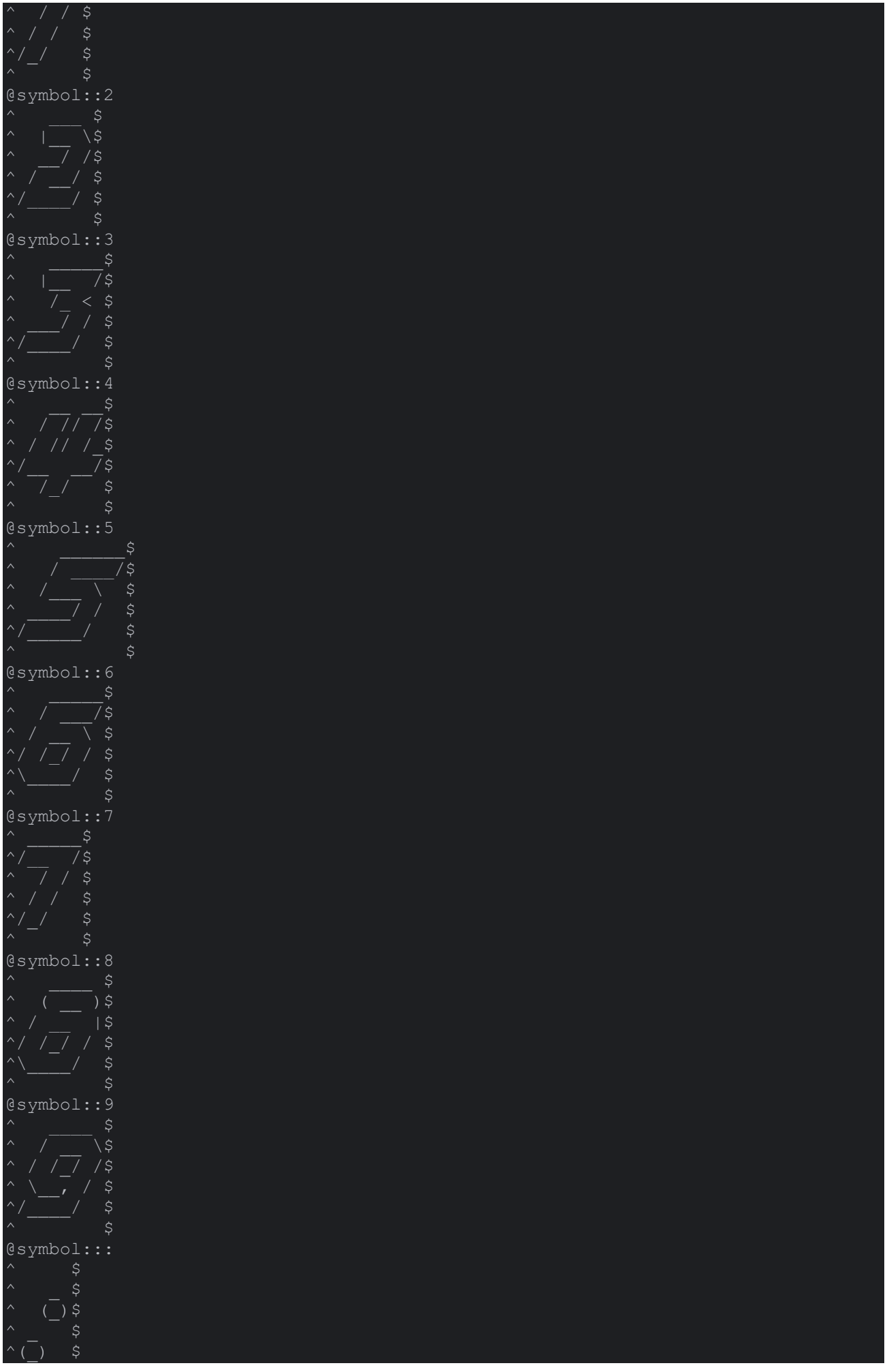
```
                        result[k + i * 6] = representation[k]

        return Fore.__getattribute__(colors[color_position]) + reduce(lambda x,
y: x + "\n" + y, result.values())
```

fori.txt:

```
height::6
@data
@symbol::
^       $
^       $
^       $
^       $
^       $
^       $
@symbol::!
^      //$
^     // $
^    //  $
^   //   $
^  //    $
^.       $
@symbol::"
^( | )$
^|/|/ $
^      $
^      $
^      $
^      $
@symbol::#
^       __ __  $
^   __/ // /_$
^ /_  _  __/$
^/_  _  __/ $
^ /_//_/    $
^           $
@symbol::$
^      __$
^    _/ /$
^   / __/$
^ (_  ) $
^/  _/  $
^/_/    $
@symbol::%
^     _   __$
^   (_)_/_/$
^    _/_/  $
^  _/_/_   $
^/_/ (_)   $
^          $
@symbol::&
^     ___    $
^   ( _ )    $
^ /  __ \/|$
^/ /_/  < $
^\____/\/ $
^         $
@symbol::&
^    __ $
^ (  )$
^ | / $
^ |/  $
^ .   $
^     $
```

```
@symbol::(
^     __ $
^   _/ _/ $
^  / /  $
^ / /   $
^/ /    $
^|_|    $
@symbol::)
^       _ $
^     | |$
^     / /$
^    / / $
^  _/_/  $
^/_/    $
@symbol::*
^       $
^  __/|_$
^ |    /$
^/_ __| $
^ |/    $
^       $
@symbol::+
^       $
^      __ $
^ __/ /_$
^/_   __/$
^ /_/    $
^       $
@symbol::,
^    $
^    $
^    $
^  _ $
^( )$
^|/ $
@symbol::-
^       $
^       $
^ ______$
^/_____/$
^  $    $
^       $
@symbol::.
^    $
^    $
^    $
^  _ $
^(_)$
^    $
@symbol::/
^         __$
^      _/_/$
^    _/_/  $
^  _/_/    $
^/_/      $
^         $
@symbol::0
^    ____ $
^  /  __ \$
^ / / / /$
^/ /_/ / $
^\____/  $
^        $
@symbol::1
^     ___$
^  <  /$
```


^ / / $
^ / / $
^/_/ $
^ $
@symbol::2
^   ___ $
^ |__ \$
^   __/ /$
^ / __/ $
^/____/ $
^ $
@symbol::3
^   _____$
^ |__ /$
^   /_ < $
^ ___/ / $
^/____/ $
^ $
@symbol::4
^   __ __$
^   / // /$
^ / // /_$
^/__ __/$
^ /_/ $
^ $
@symbol::5
^   ______$
^   / ____/$
^   /___ \ $
^ ____/ / $
^/_____/ $
^ $
@symbol::6
^   _____$
^   / ___/$
^ / __ \ $
^/ /_/ / $
^____/ $
^ $
@symbol::7
^ _____$
^/__ /$
^ / / $
^ / / $
^/_/ $
^ $
@symbol::8
^   ____ $
^   (__)$
^ / __ |$
^/ /_/ / $
^____/ $
^ $
@symbol::9
^   ____ $
^   / __ \$
^ / /_/ /$
^ __, / $
^/____/ $
^ $
@symbol:::
^ $
^ $
^   (_)$
^ _ $
^(_) $

```
^      $
@symbol::;
^      $
^     _ $
^   (_)$
^      $
^ _    $
^( )   $
^|/    $
@symbol::<
^    __$
^  / /$
^/ / $
^\ \ $
^ \_\$
^     $
@symbol::=
^         $
^  ______$
^ /_____/$
^/_____/ $
^   $     $
^         $
@symbol::>
^ __    $
^\ \ $
^ \ \$
^ / /$
^/_/ $
^     $
@symbol::?
^  ____ $
^ /__ \$
^   / _/$
^ /_/  $
^(_)   $
^      $
@symbol::@
^     _______ $
^   / ____ \$
^ / / ___ `/$
^/ / /_/ / $
^\ \__,_/  $
^ \_____/  $
@symbol::A
^      ___ $
^    /   |$
^   / /| |$
^  / ___ |$
^/_/  |_|$
^       $
@symbol::B
^      ____ $
^    / __ )$
^   / __  |$
^  / /_/ / $
^/_____/  $
^       $
@symbol::C
^     ______$
^   / ____/$
^  / /     $
^/ /___   $
^\____/   $
^         $
@symbol::D
^      ____ $
```

```
^    /   __ \$
^   / / / /$
^  / /_/ / $
^/_____/  $
^         $
@symbol::E
^         _____$
^    /   ____/$
^   / __/     $
^  / /___     $
^/_____/     $
^            $
@symbol::F
^         _____$
^    /   ____/$
^   / /_       $
^  / __/       $
^/_/          $
^             $
@symbol::G
^      _____$
^   /  ____/$
^  / /  __  $
^/ /_/ /    $
^\____/     $
^           $
@symbol::H
^         __  __$
^    /   / / / /$
^   / /_/ / /  $
^  / __  /     $
^/_/ /_/      $
^             $
@symbol::I
^         ____$
^    /   _/$
^    / /    $
^  _/ /     $
^/___/     $
^          $
@symbol::J
^           __$
^        / /$
^   __  / /  $
^/ /_/ /    $
^\____/     $
^           $
@symbol::K
^         __ __$
^    /   //_/$
^   / ,<     $
^  / /| |    $
^/_/ |_|    $
^           $
@symbol::L
^         __  $
^    /   / /  $
^   / /    $
^  / /___  $
^/_____/$
^         $
@symbol::M
^            __  ___$
^    /   |/  /$
^   / /|_/ / /  $
^  / /  / /   $
```

```
^/_/  /_/   $
^          $
@symbol::N
^      _    __$
^    / |  / /$
^   /  |/ / $
^  / /|  /  $
^ /_/ |_/   $
^          $
@symbol::O
^      ____ $
^    / __ \$
^  / / / /$
^ / /_/ / $
^ \____/  $
^         $
@symbol::P
^      ____ $
^    / __ \$
^   / /_/ /$
^  / ____/ $
^ /_/      $
^          $
@symbol::Q
^      ____ $
^    / __ \$
^  / / / /$
^ / /_/ / $
^ \___\_\ $
^         $
@symbol::R
^      ____ $
^    / __ \$
^   / /_/ /$
^  / _, _/ $
^ /_/ |_|  $
^          $
@symbol::S
^     _____$
^   / ___/$
^   \__ \ $
^  ___/ / $
^ /____/  $
^         $
@symbol::T
^  _______$
^ /_  __/$
^  / /    $
^ / /     $
^/_/      $
^         $
@symbol::U
^     __  __$
^   / / / /$
^  / / / /  $
^ / /_/ /   $
^ \____/    $
^           $
@symbol::V
^ _      __$
^| |  / /$
^| | / / $
^| |/ /  $
^|___/   $
^        $
@symbol::W
```

```
^ _          __$
^| |      / /$
^| | /| / / $
^| |/ |/ /  $
^|__/|__/   $
^           $
@symbol::X
^    _   __$
^   | |/ /$
^   |   / $
^ /    |  $
^/_/|_|  $
^        $
@symbol::Y
^__    __$
^\ \/ /$
^ \  / $
^ / /  $
^/_/   $
^      $
@symbol::Z
^ _____$
^/___ /$
^  / / $
^ / /__$
^/____/$
^       $
@symbol::[
^       ___$
^     /  _/$
^    / /  $
^   / /   $
^ / /     $
^/__/     $
@symbol::\
^ __      $
^\ \      $
^ \ \     $
^  \ \    $
^   \_\   $
^         $
@symbol::]
^       ___$
^     /  /$
^     / / $
^    / /  $
^ _/ /    $
^/__/     $
@symbol::^
^  //|$
^ |/||$
^  $  $
^ $   $
^$    $
^     $
@symbol::_
^        $
^        $
^        $
^        $
^ _______$
^/_____/$
@symbol::`
^   _ $
^ ( )$
^  V $
```

```
^ $  $
^$   $
^    $
@symbol::a
^        $
^  _____ _$
^ / __  `/$
^/ /_/ / $
^\__,_/  $
^        $
@symbol::b
^        $
^    / /_ $
^   / __ \$
^ / /_/ /$
^/_.___/ $
^        $
@symbol::c
^        $
^  _____$
^ / ___/$
^/ /__  $
^\___/  $
^        $
@symbol::d
^         __$
^  _____/ /$
^ / __  / $
^/ /_/ /  $
^\__,_/   $
^         $
@symbol::e
^       $
^  ____ $
^ / _ \$
^/  __/$
^\___/ $
^       $
@symbol::f
^     ____$
^    / __/$
^   / /_  $
^ / __/  $
^/_/     $
^        $
@symbol::g
^          $
^  ____ _$
^ / __ `/$
^ / /_/ / $
^ \__, /  $
^/____/   $
@symbol::h
^        $
^    / /_ $
^   / __ \$
^ / / / /$
^/_/ /_/ $
^        $
@symbol::i
^      _ $
^     (_)$
^    / / $
^   / /  $
^/_/   $
^      $
```

```
@symbol::j
^        _ $
^       (_)$
^      / / $
^     / /  $
^  __/ /   $
^/____/    $
@symbol::k
^      __  $
^    / /__$
^   / //_/$
^  / ,<   $
^/_/|_|  $
^        $
@symbol::l
^      __$
^    / /$
^   / / $
^  / /  $
^/_/   $
^      $
@symbol::m
^              $
^              $
^   _____ ____ $
^  / __  `__ \$
^ / / / / / / /$
^/_/ /_/ /_/ $
^              $
@symbol::n
^         $
^     ____ $
^   / __ \$
^  / / / /$
^/_/ /_/ $
^         $
@symbol::o
^         $
^   _____ $
^  / __ \$
^ / /_/ /$
^ \____/ $
^         $
@symbol::p
^           $
^      ____ $
^    / __ \$
^   / /_/ /$
^  / .___/ $
^ /_/      $
@symbol::q
^         $
^   _____ $
^  / ___ `/$
^ / /_/ / $
^ \__, /  $
^   /_/   $
@symbol::r
^         $
^      _____$
^   / ___/$
^  / /     $
^ /_/      $
^         $
@symbol::s
^         $
^     _____$
```

```
^   / ___/$
^  (__  ) $
^/____/  $
^        $
@symbol::t
^      $
^   __ $
^  / /_$
^ / __/$
^/ /_  $
^\__/  $
^      $
@symbol::u
^         $
^   __  __$
^ / / / / /$
^/ /_/ / $
^\__,_/  $
^        $
@symbol::v
^        $
^ _    __$
^| |  / /$
^| | / / $
^|___/  $
^       $
@symbol::w
^           $
^ _         __$
^| | /| / / /$
^| |/ |/ / $
^|__/|__/  $
^          $
@symbol::x
^         $
^    _  __$
^   | |/_/$
^  _>  <  $
^/_/|_|  $
^        $
@symbol::y
^          $
^     __  __$
^   / / / / /$
^  / /_/ / $
^  \__, /  $
^/____/   $
@symbol::z
^      $
^  ____$
^/   /$
^  / /_$
^/___/$
^      $
@symbol::{
^       __$
^    _/_/$
^  _/_/  $
^< <     $
^/ /     $
^\_\     $
@symbol::|
^ ___$
^| |$
^| |$
^| |$
^| |$
```

Висновки. Виконуючи ці завдання, студенти мають створити генератор ASCII-арту з нуля, та надати можливість налаштовувати символи, розміри, вирівнювання та кольори, що дозволить їм глибше розібратися як створюється ASCII-арт.