**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Кафедра інформаційних систем та мереж

**Звіт**

**з дисципліни «Спеціалізовані мови програмування»**

**Лабораторна робота № 3**

**Виконав:**

студент групи ІТ-21сп

Масняк Б. В.

**Прийняв**:

Щербак С.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2023

**Мета:** створення додатка Генератора ASCII-арту.

**Хід роботи**

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).

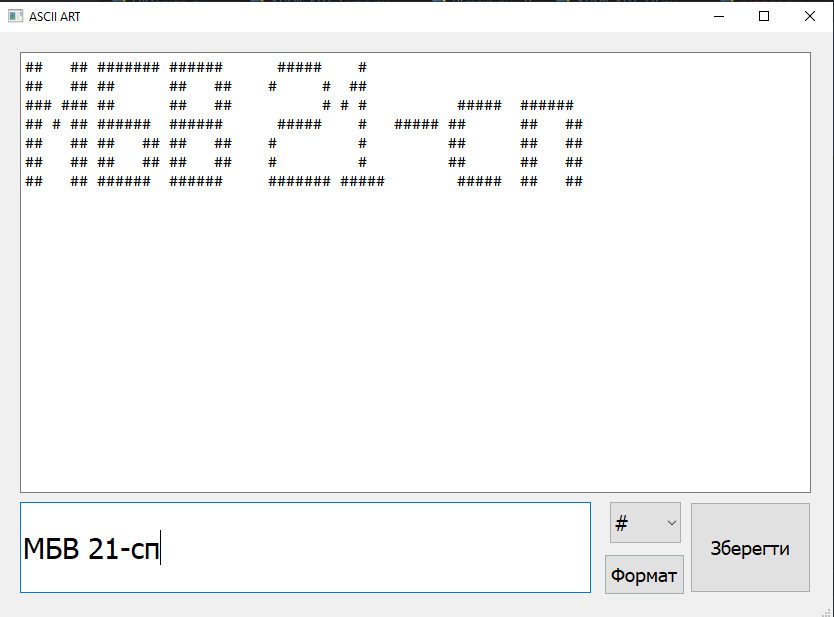
Завдання 9: Функція попереднього перегляду

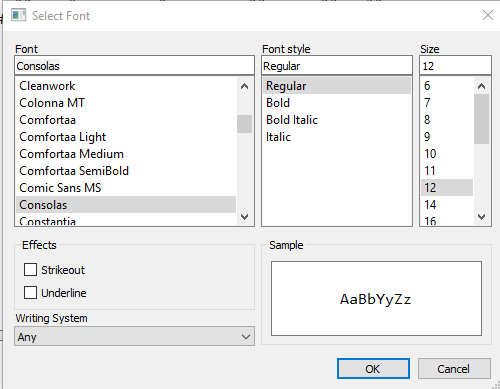
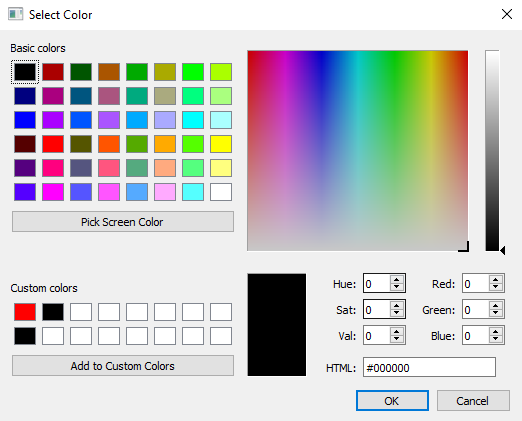
Реалізуйте функцію попереднього перегляду, яка показує користувачам попередній перегляд їхнього ASCII-арту перед остаточним збереженням.

Завдання 10: Інтерфейс, зрозумілий для користувача

Створіть зручний для користувача інтерфейс командного рядку для додатка, щоб зробити його інтуїтивно зрозумілим та легким у використанні.

|  |
| --- |
| **main.py**  import ASCII\_ART\_UI  if \_\_name\_\_ == "\_\_main\_\_":  while True:  ASCII\_ART\_UI.\_init\_() |
| **ASCII\_ART\_UI.py**  import sys  import ASCII\_ART\_Func  from PyQt5 import QtCore, QtGui, QtWidgets  def \_init\_():  class Ui\_MainWindow(object):  def setupUi(self, MainWindow):  MainWindow.setObjectName("MainWindow")  MainWindow.resize(833, 587)  self.centralwidget = QtWidgets.QWidget(MainWindow)  self.centralwidget.setObjectName("centralwidget")  self.textBrowser = QtWidgets.QTextBrowser(self.centralwidget)  self.textBrowser.setGeometry(QtCore.QRect(20, 20, 791, 441))  self.textBrowser.setObjectName("textBrowser")  self.textBrowser.setFont(QtGui.QFont("Consolas", 12))  self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)  self.lineEdit.setGeometry(QtCore.QRect(20, 470, 571, 91))  self.lineEdit.setObjectName("lineEdit")  font = QtGui.QFont()  font.setPointSize(22)  self.lineEdit.setFont(font)  self.saveButton = QtWidgets.QPushButton(self.centralwidget)  self.saveButton.setGeometry(QtCore.QRect(690, 470, 121, 91))  font = QtGui.QFont()  font.setPointSize(14)  self.saveButton.setFont(font)  self.saveButton.setObjectName("saveButton")  self.symbolcomboBox = QtWidgets.QComboBox(self.centralwidget)  self.symbolcomboBox.setGeometry(QtCore.QRect(610, 470, 71, 41))  font = QtGui.QFont()  font.setPointSize(16)  self.symbolcomboBox.setFont(font)  self.symbolcomboBox.setObjectName("symbolcomboBox")  self.symbolcomboBox.addItem("")  self.symbolcomboBox.addItem("")  self.symbolcomboBox.addItem("")  self.formatButton = QtWidgets.QPushButton(self.centralwidget)  self.formatButton.setGeometry(QtCore.QRect(604, 522, 81, 41))  font = QtGui.QFont()  font.setPointSize(14)  self.formatButton.setFont(font)  self.formatButton.setObjectName("formatButton")  MainWindow.setCentralWidget(self.centralwidget)  self.statusbar = QtWidgets.QStatusBar(MainWindow)  self.statusbar.setObjectName("statusbar")  MainWindow.setStatusBar(self.statusbar)  self.retranslateUi(MainWindow)  QtCore.QMetaObject.connectSlotsByName(MainWindow)  self.addFunc()  def retranslateUi(self, MainWindow):  \_translate = QtCore.QCoreApplication.translate  MainWindow.setWindowTitle(\_translate("MainWindow", "ASCII ART"))  self.saveButton.setText(\_translate("MainWindow", "Зберегти"))  self.symbolcomboBox.setItemText(0, \_translate("MainWindow", "#"))  self.symbolcomboBox.setItemText(1, \_translate("MainWindow", "\*"))  self.symbolcomboBox.setItemText(2, \_translate("MainWindow", "@"))  self.formatButton.setText(\_translate("MainWindow", "Формат"))  def addFunc(self):  self.lineEdit.textChanged.connect(lambda: ASCII\_ART\_Func.update\_text\_browser(self.textBrowser, self.lineEdit, self.symbolcomboBox))  self.saveButton.clicked.connect(lambda: ASCII\_ART\_Func.saveFunc(self.textBrowser))  self.symbolcomboBox.activated.connect(lambda: ASCII\_ART\_Func.update\_text\_browser(self.textBrowser, self.lineEdit, self.symbolcomboBox))  self.formatButton.clicked.connect(lambda: ASCII\_ART\_Func.formatFunc(self.textBrowser, self.lineEdit, self.symbolcomboBox))  app = QtWidgets.QApplication(sys.argv)  MainWindow = QtWidgets.QMainWindow()  ui = Ui\_MainWindow()  ui.setupUi(MainWindow)  MainWindow.show()  sys.exit(app.exec\_()) |
| **ASCII\_ART\_Func .py**  from pyfiglet import Figlet  from PyQt5.QtWidgets import QFontDialog, QColorDialog  def update\_text\_browser(textBrowser, lineEdit, symbolcomboBox):  f = Figlet(font="banner")  text = f.renderText(lineEdit.text())  text = text.replace("#", symbolcomboBox.currentText())  textBrowser.setPlainText(text)  def saveFunc(textBrowser):  format = textBrowser.currentCharFormat()  font = format.font()  color = format.foreground().color()  with (open("save.html", "w") as html\_file):  txt\_content = textBrowser.toPlainText()  css\_content = ".text-block {\n"+f" font-family: {font.family()};\n"+f" font-size: {font.pointSize()}pt;\n"+f" color: {color.name()};\n"+"}"  html\_content = f"""  <!DOCTYPE html>  <html>  <head>  <style>  {css\_content}  </style>  <link href="style.css" rel="stylesheet" type="text/css" />  <meta charset="UTF-8">  <title>Save</title>  </head>  <body>  <div class="text-block"><pre>  {txt\_content}  </pre>  </div>  </body>  </html>  """  html\_file.write(html\_content)  with open('style.css', 'w') as css\_file:  css\_file.write(".text-block {\n")  css\_file.write(f" font-family: {font.family()};\n")  css\_file.write(f" font-size: {font.pointSize()}pt;\n")  css\_file.write(f" color: {color.name()};\n")  css\_file.write("}\n")  with open('save.txt', 'w') as fileSave:  fileSave.write(textBrowser.toPlainText())  def formatFunc(textBrowser, lineEdit, symbolcomboBox):  font, ok = QFontDialog.getFont(textBrowser.currentFont())  color = QColorDialog.getColor(initial=textBrowser.textColor(), parent=textBrowser)  if ok:  textBrowser.setFont(font)  if color.isValid():  textBrowser.setTextColor(color)  update\_text\_browser(textBrowser, lineEdit, symbolcomboBox) |



**Висновок:** під час виконання цієї лабораторної роботи я створив універсальний Генератор ASCII-арту, який дозволить користувачам налаштовувати свої творіння з різними шрифтами, кольорами, розмірами та символами. Проект надасть практичний досвід роботи з введенням користувача, зовнішніми бібліотеками, роботою з файлами та дизайном інтерфейсу користувача в Python.