Міністерство освіти і науки України

**Прикарпатський національний університет**

**імені В.Стефаника**

*Факультет математики та інформатики*

*Кафедра інформаційних технологій*

*Людинно-машинна взаємодія*

Лабораторна робота № 6

Тема: Контейнерний клас QVector. Робота з двовимірними масивами

*Варіант 2*

Виконав: ***Гук Д.П.***

Група ІПЗ-31

Дата:20 жовтня 2023 р.

Викладач: Пікуляк М.В.

Івано-Франківськ – 2023

**Мета роботи:**

Отримати навички роботи з двовимірними масивами в ICP “Qt-Creator”.

**Завдання для виконання :**

1. Дана матриця розміру m x n. Знайти суми елементів всіх її парних і

непарних стовпців (знаходження суми парних / непарних стовпців

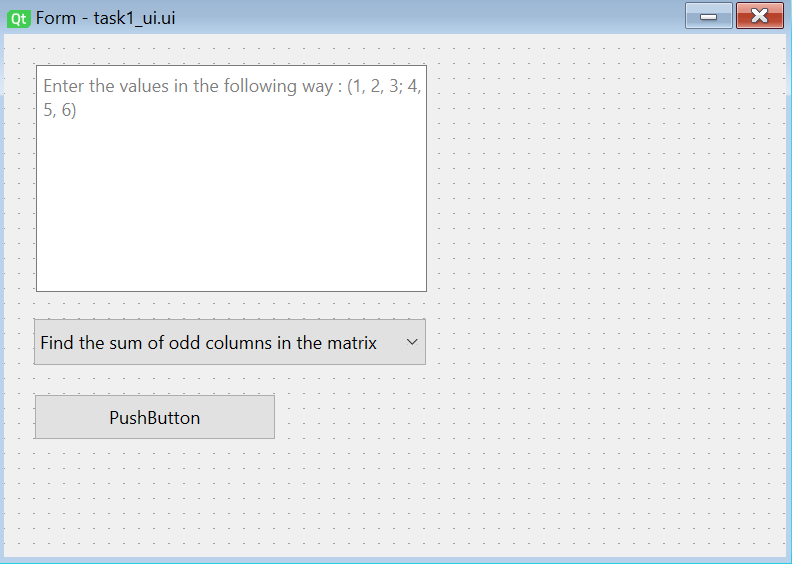
визначається користувачем у віджеті comboBox).

2. Дана матриця розміру m x n. Перетворити матрицю, помінявши місцями

мінімальний і максимальний елемент в кожному рядку.

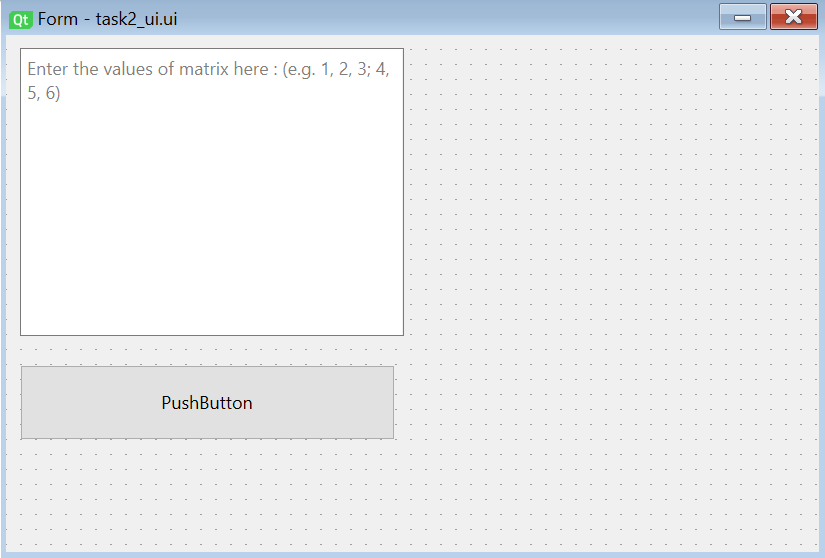
**Тексти скриптів і зображення діалогових вікон QtCreator з виконаними завданнями :**

**Завдання №1:**



from PyQt6 import QtCore, QtGui, QtWidgets  
import sys  
  
class Ui\_Form(object):  
 def setupUi(self, Form):  
 Form.setObjectName("Task 1")  
 Form.resize(521, 349)  
 self.matrix\_input = QtWidgets.QPlainTextEdit(parent=Form)  
 self.matrix\_input.setGeometry(QtCore.QRect(21, 21, 261, 151))  
 self.matrix\_input.setObjectName("plainTextEdit")  
 self.action\_selection = QtWidgets.QComboBox(parent=Form)  
 self.action\_selection.setGeometry(QtCore.QRect(20, 190, 261, 31))  
 self.action\_selection.setObjectName("comboBox")  
 self.action\_selection.addItem("")  
 self.action\_selection.addItem("")  
 self.pushButton = QtWidgets.QPushButton(parent=Form)  
 self.pushButton.setGeometry(QtCore.QRect(20, 240, 161, 31))  
 self.pushButton.setObjectName("pushButton")  
 self.result\_label = QtWidgets.QLabel(parent=Form)  
 self.result\_label.setGeometry(QtCore.QRect(20, 300, 251, 31))  
 self.result\_label.setText("")  
 self.result\_label.setObjectName("label")  
  
 self.retranslateUi(Form)  
 QtCore.QMetaObject.connectSlotsByName(Form)  
  
 self.pushButton.clicked.connect(self.calculate)  
  
 def retranslateUi(self, Form):  
 \_translate = QtCore.QCoreApplication.translate  
 Form.setWindowTitle(\_translate("Form", "Task 1"))  
 self.matrix\_input.setPlaceholderText(\_translate("Form", "Enter the values in the following way : \n(\ne.g.\n1, 2, 3; \n4, 5, 6\n)"))  
 self.action\_selection.setItemText(0, \_translate("Form", "Find the sum of even columns in the matrix"))  
 self.action\_selection.setItemText(1, \_translate("Form", "Find the sum of odd columns in the matrix"))  
 self.pushButton.setText(\_translate("Form", "Calculate the result"))  
  
 def calculate(self):  
 try:  
 matrix\_str = self.matrix\_input.toPlainText()  
 matrix = [[int(num) for num in row.split(',')] for row in matrix\_str.split(';')]  
  
 if len(matrix) < 1 or any(len(row) != len(matrix[0]) for row in matrix):  
 self.result\_label.setText("Invalid matrix format.")  
 return  
  
 if self.action\_selection.currentIndex() == 0:  
 even\_sum = self.calculate\_even\_columns(matrix)  
 self.result\_label.setText(f"Sum of even columns: {even\_sum}")  
 elif self.action\_selection.currentIndex() == 1:  
 odd\_sum = self.calculate\_odd\_columns(matrix)  
 self.result\_label.setText(f"Sum of odd columns: {odd\_sum}")  
  
 except Exception as e:  
 self.result\_label.setText(f"Error: {e}")  
  
 def calculate\_even\_columns(self, matrix):  
 even\_column\_sum = 0  
 for j in range(len(matrix[0])):  
 if j % 2 == 0:  
 for i in range(len(matrix)):  
 even\_column\_sum += matrix[i][j]  
 return even\_column\_sum  
  
 def calculate\_odd\_columns(self, matrix):  
 odd\_column\_sum = 0  
 for j in range(len(matrix[0])):  
 if j % 2 != 0:  
 for i in range(len(matrix)):  
 odd\_column\_sum += matrix[i][j]  
 return odd\_column\_sum  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QtWidgets.QApplication(sys.argv)  
 task1\_dialog = QtWidgets.QDialog()  
 ui = Ui\_Form()  
 ui.setupUi(task1\_dialog)  
 task1\_dialog.show()  
 sys.exit(app.exec())

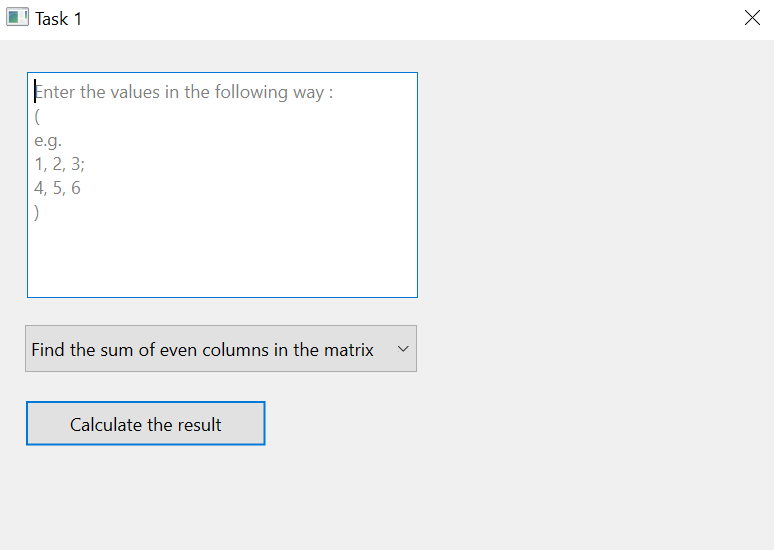
**Завдання №2:**

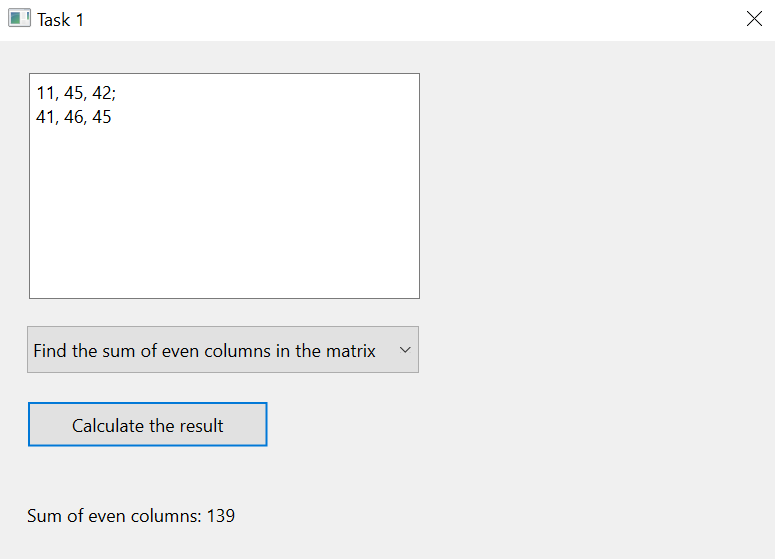
****

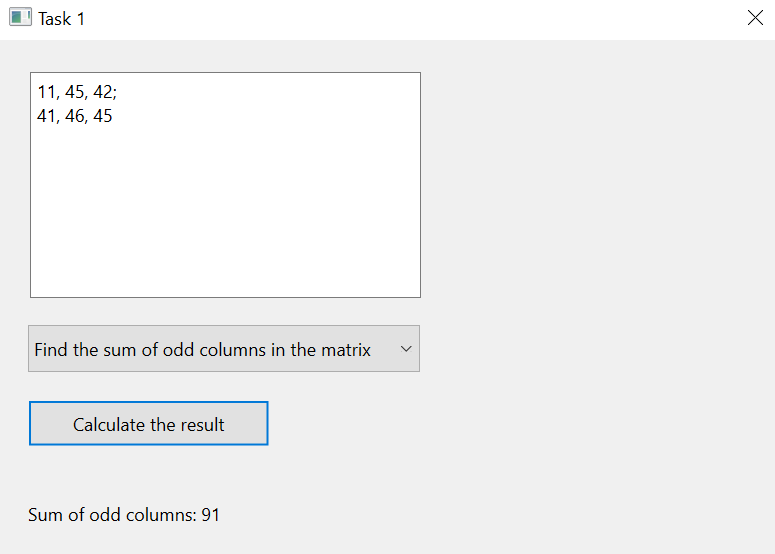
from PyQt6 import QtCore, QtGui, QtWidgets  
import sys  
  
class Ui\_Form(object):  
 def setupUi(self, Form):  
 Form.setObjectName("Task 2")  
 Form.resize(542, 345)  
 self.matrix\_input = QtWidgets.QTextEdit(parent=Form)  
 self.matrix\_input.setGeometry(QtCore.QRect(9, 9, 256, 192))  
 self.matrix\_input.setObjectName("textEdit")  
 self.transform\_button = QtWidgets.QPushButton(parent=Form)  
 self.transform\_button.setGeometry(QtCore.QRect(9, 220, 250, 50))  
 self.transform\_button.setObjectName("pushButton")  
 self.result\_label = QtWidgets.QLabel(parent=Form)  
 self.result\_label.setGeometry(QtCore.QRect(9, 280, 250, 60))  
 self.result\_label.setText("")  
 self.result\_label.setObjectName("label")  
  
 self.transform\_button.clicked.connect(self.transform) # Connect button click to transform function  
  
 self.retranslateUi(Form)  
 QtCore.QMetaObject.connectSlotsByName(Form)  
  
 def retranslateUi(self, Form):  
 \_translate = QtCore.QCoreApplication.translate  
 Form.setWindowTitle(\_translate("Form", "Task 2"))  
 self.matrix\_input.setPlaceholderText(\_translate("Form", "Enter the values in the following way : \n(\ne.g.\n1, 2, 3; \n4, 5, 6\n)"))  
 self.transform\_button.setText(\_translate("Form", "Transform the matrix"))  
  
 def transform(self):  
 try:  
 matrix\_str = self.matrix\_input.toPlainText()  
 matrix = [[int(num) for num in row.split(',')] for row in matrix\_str.split(';')]  
  
 if len(matrix) < 1 or any(len(row) != len(matrix[0]) for row in matrix):  
 self.result\_label.setText("Invalid matrix format.")  
 return  
  
 transformed\_matrix = self.transform\_matrix(matrix)  
 self.result\_label.setText("Transformed Matrix:")  
 for row in transformed\_matrix:  
 self.result\_label.setText(self.result\_label.text() + f"\n{', '.join(map(str, row))}")  
  
 except Exception as e:  
 self.result\_label.setText(f"Error: {e}")  
  
 def swap\_min\_max(self, row):  
 min\_val = min(row)  
 max\_val = max(row)  
 min\_idx = row.index(min\_val)  
 max\_idx = row.index(max\_val)  
 row[min\_idx], row[max\_idx] = max\_val, min\_val  
 return row  
  
 def transform\_matrix(self, matrix):  
 transformed\_matrix = []  
 for row in matrix:  
 transformed\_row = self.swap\_min\_max(row)  
 transformed\_matrix.append(transformed\_row)  
 return transformed\_matrix  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QtWidgets.QApplication(sys.argv)  
 task2\_dialog = QtWidgets.QDialog()  
 ui = Ui\_Form()  
 ui.setupUi(task2\_dialog)  
 task2\_dialog.show()  
 sys.exit(app.exec())

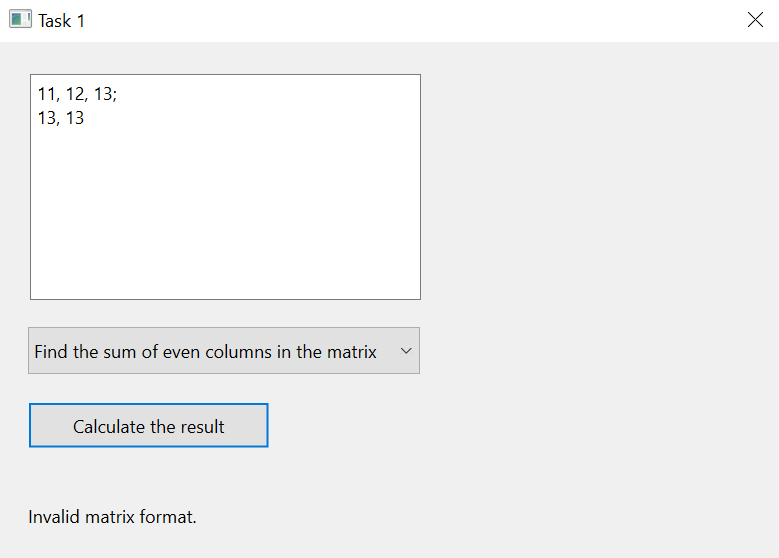
**Cкрін-шоти виконання завдань лабораторної роботи :**

**Завдання №1 :**

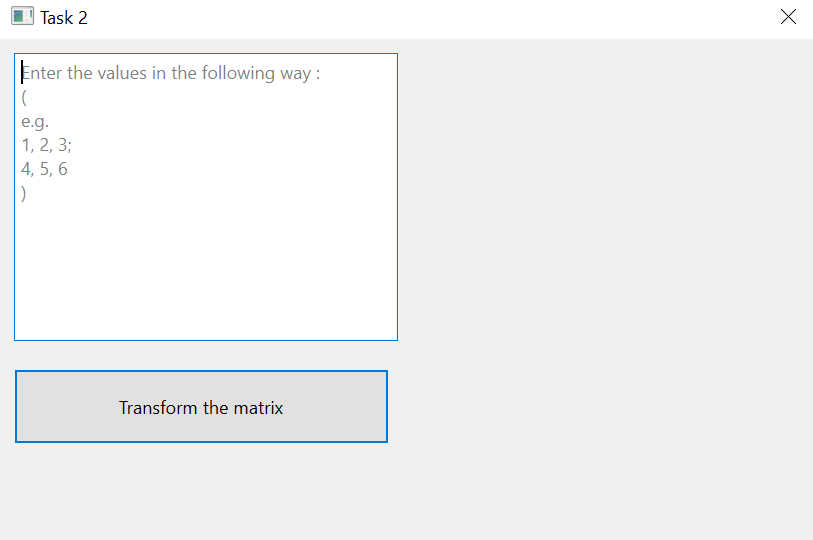
****

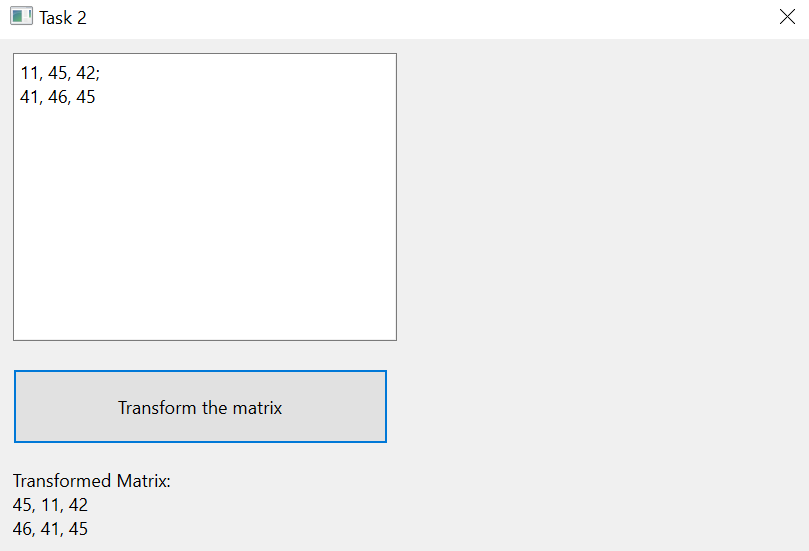
****

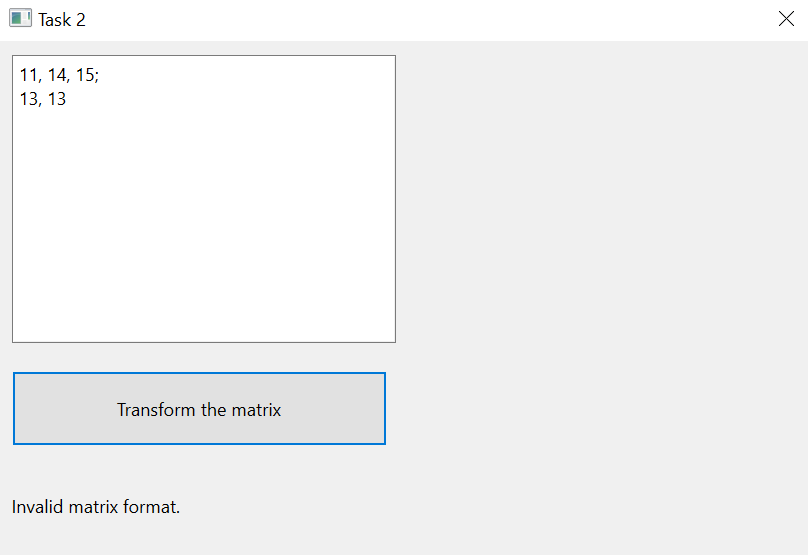
****

****

**Завдання №2 :**

****

****

****