 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №6**

з дисципліни «Теорія прийняття рішень»

на тему: «ДИНАМІЧНЕ ПРОГРАМУВАННЯ»

**Виконав:**

студент гр. БС-93

Шкепаст М. В.

**Перевірила:**

Піднебесна Г. А.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

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

(підпис викладача)

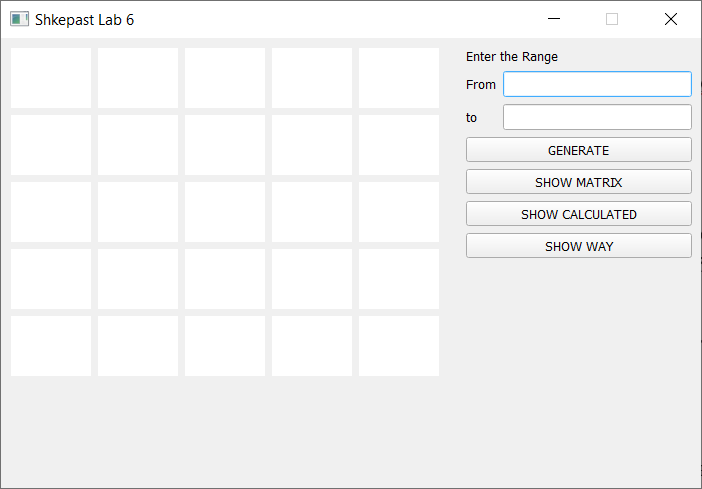
Київ-2022

**Мета  роботи**: навчитись використовувати метод динамічного програмування для вирішення практичних задач.

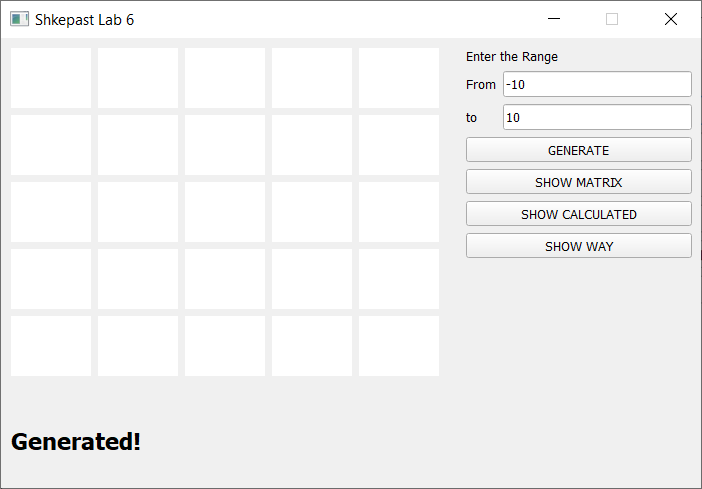
**Виконання:**

Програмний застосунок до лабораторної роботи був написаний мовою Python, графічний інтерфейс створений за допомогою бібліотек NumPy, PyQt5.

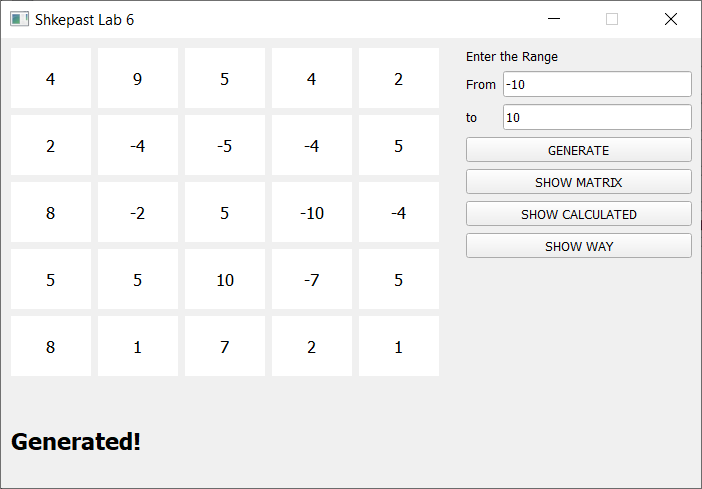
При запуску ПЗ треба ввести у відповідні поля «From» і «to» границі псевдовипадкових чисел, якими буде заповнена матриця 5х5:



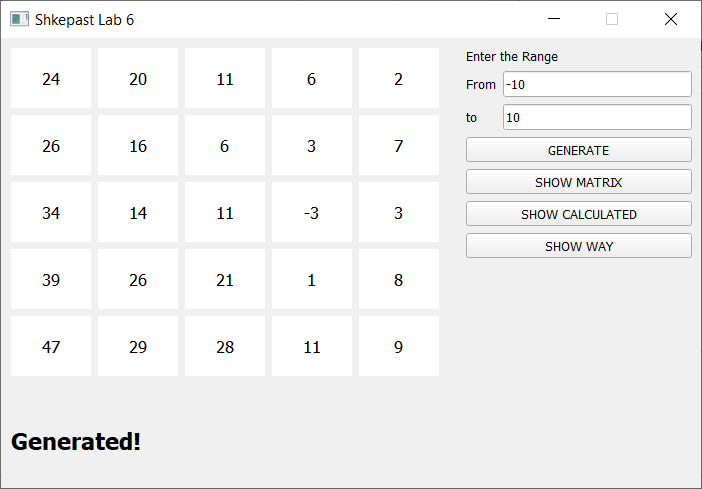
Після вводу границь треба натиснути кнопку «GENERATE» і якщо все введено вірно, знизу побачимо напис що свідчить про те, що матриця згенерована:



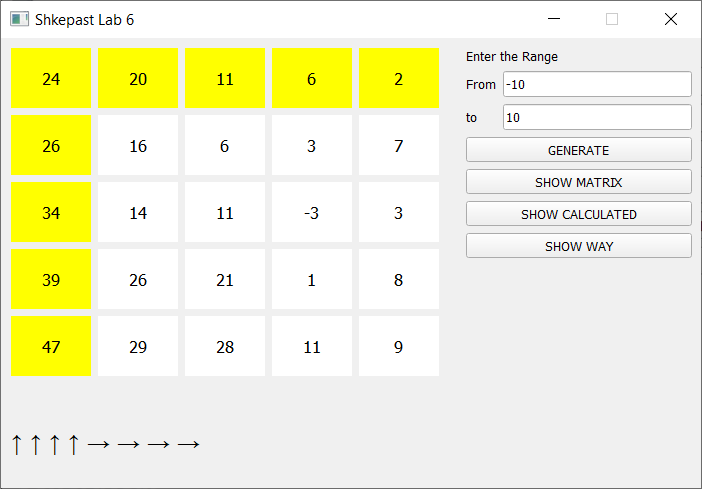
Для того щоб побачити згенеровану матрицю, натискаємо кнопку «SHOW MATRIX»:



Натиснувши на кнопку «SHOW CALCULATED» побачимо матрицю значень комірок, розрахованих за допомогою динамічного програмування:



Натиснувши на кнопку «», побачимо оптимальний шлях, яким можно пройти від лівої нижньої комірки, до правої верхньої, пересуваючись лише вправо, вгору або по діагоналі:



**Висновок:**

У даній роботі ми набули практичних навичок у застосуванні методу динамічного програмування для вирішення практичних задач. Написавши програмний застосунок ми змогли отримати розв’язок поставленої задачі.

**Лістинг**

from random import randint  
  
from PyQt5 import Qt  
from PyQt5.QtWidgets import \*  
from PyQt5.QtCore import Qt  
  
import numpy as np  
import mainui  
  
  
def calc(matrix, test\_matrix, row, col):  
 return matrix[row][col] + max(test\_matrix[row-1][col], test\_matrix[row][col+1], test\_matrix[row-1][col+1])  
  
  
def choose\_way(matrix, row, col):  
 if row > 0:  
 uno = matrix[row - 1][col]  
 else:  
 return 1  
 if col < -1:  
 dos = matrix[row][col + 1]  
 else:  
 return 0  
 tres = matrix[row - 1][col + 1]  
 return np.argmax([uno, dos, tres])  
  
  
class MatplotlibWidget(QMainWindow):  
 def \_\_init\_\_(self):  
 super(MatplotlibWidget, self).\_\_init\_\_()  
 self.way = None  
 self.ui = mainui.Ui\_Form()  
 self.ui.setupUi(self)  
 self.tmp = None  
 self.tmp\_test = None  
  
 self.ui.pushButton\_generate.clicked.connect(self.generate)  
 self.ui.pushButton\_show\_way.clicked.connect(self.show\_way)  
 self.ui.pushButton\_show\_matrix.clicked.connect(self.show\_matrix)  
 self.ui.pushButton\_show\_calculated.clicked.connect(self.show\_calculated)  
  
 self.labels = [  
 [self.ui.label\_idx\_0\_0, self.ui.label\_idx\_0\_1, self.ui.label\_idx\_0\_2, self.ui.label\_idx\_0\_3,  
 self.ui.label\_idx\_0\_4],  
 [self.ui.label\_idx\_1\_0, self.ui.label\_idx\_1\_1, self.ui.label\_idx\_1\_2, self.ui.label\_idx\_1\_3,  
 self.ui.label\_idx\_1\_4],  
 [self.ui.label\_idx\_2\_0, self.ui.label\_idx\_2\_1, self.ui.label\_idx\_2\_2, self.ui.label\_idx\_2\_3,  
 self.ui.label\_idx\_2\_4],  
 [self.ui.label\_idx\_3\_0, self.ui.label\_idx\_3\_1, self.ui.label\_idx\_3\_2, self.ui.label\_idx\_3\_3,  
 self.ui.label\_idx\_3\_4],  
 [self.ui.label\_idx\_4\_0, self.ui.label\_idx\_4\_1, self.ui.label\_idx\_4\_2, self.ui.label\_idx\_4\_3,  
 self.ui.label\_idx\_4\_4]]  
  
 def show\_matrix(self):  
 try:  
 for row in range(len(self.tmp)):  
 for col in range(len(self.tmp)):  
 self.labels[row][col].setText(str(int(self.tmp[row][col])))  
 self.labels[row][col].setAlignment(Qt.AlignCenter)  
 except:  
 self.ui.label\_route.setText('First you need to generate matrix')  
 return 0  
  
 def show\_calculated(self):  
 try:  
 for row in range(len(self.tmp\_test)):  
 for col in range(len(self.tmp\_test)):  
 self.labels[row][col].setText(str(int(self.tmp\_test[row][col])))  
 self.labels[row][col].setAlignment(Qt.AlignCenter)  
 except:  
 self.ui.label\_route.setText('First you need to generate matrix')  
 return 0  
  
 def clear\_matrix(self):  
 for row in range(len(self.labels)):  
 for col in range(len(self.labels)):  
 self.labels[row][col].setText(' ')  
 self.labels[row][col].setAlignment(Qt.AlignCenter)  
 self.labels[row][col].setStyleSheet('background-color: rgb(255, 255, 255);')  
  
 def generate(self):  
 self.clear\_matrix()  
 try:  
 A = self.ui.lineEdit\_from.text()  
 B = self.ui.lineEdit\_to.text()  
 A = int(A)  
 B = int(B)  
 self.ui.label\_route.setText('Generated!')  
 except ValueError:  
 self.ui.label\_route.setText('Range must be int and not empty')  
 return 0  
 if A < B:  
 self.ui.label\_route.setText('Generated!')  
 else:  
 self.ui.label\_route.setText('From value must be less than To')  
 return 0  
 self.tmp = np.zeros([5, 5])  
 self.tmp\_test = np.zeros([5, 5])  
  
 for row in range(5):  
 for col in range(5):  
 self.tmp[row][col] = randint(A, B)  
  
 len\_ = len(self.tmp\_test)  
 self.tmp\_test[0][-1] = self.tmp[0][-1]  
 for row in range(1, len\_):  
 self.tmp\_test[row][-1] = self.tmp[row][-1] + self.tmp\_test[row - 1][-1]  
 for col in range(1, len\_):  
 self.tmp\_test[0][-1 - col] = self.tmp[0][-1 - col] + self.tmp\_test[0][-col]  
 for n in range(1, len\_):  
 for row in range(n, len\_):  
 self.tmp\_test[row][-1 - n] = calc(self.tmp, self.tmp\_test, row, -1 - n)  
 for col in range(n, len\_):  
 self.tmp\_test[n][-1 - col] = calc(self.tmp, self.tmp\_test, n, -1 - col)  
  
 def show\_way(self):  
 try:  
 ways = ['↑', '→', '↗']  
 self.way = ''  
 row, col = 4, -5  
 while row != 0 or col != -1:  
 idx = choose\_way(self.tmp\_test, row, col)  
 self.way += ways[idx] + ' '  
 if idx == 0:  
 if row != 0:  
 row -= 1  
 else:  
 col += 1  
 elif idx == 1:  
 if col != 0:  
 col += 1  
 else:  
 row -= 1  
 elif idx == 2:  
 row -= 1  
 col += 1  
 self.labels[row][col].setStyleSheet('background-color: rgb(255, 255, 0);')  
 route = self.way.split(' ')[:-1]  
 self.ui.label\_route.setText(self.way)  
 self.ui.label\_idx\_4\_0.setStyleSheet('background-color: rgb(255, 255, 0);')  
 except:  
 self.ui.label\_route.setText('First you need to generate matrix')  
 return 0  
  
  
app = QApplication([])  
  
app.setStyle('Fusion')  
window = MatplotlibWidget()  
  
window.setWindowTitle('Shkepast Lab 6')  
window.show()  
app.exec\_()