Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Лабораторна робота №8

з дисципліни «Паралельне програмування -1.

Основи паралельного програмування»

Тема: **ПОТОКИ В МОВІ PYTHON**

Виконав

студент групи ІП-31

Кобилинський Дмитро

Київ 2015

**Цель работы:** изучение средств языка Python для работы с задачами (процессами).

**Выполнение работы:** Разработать программу, содержащую п а р а л л е л ь н ы е потоки,

каждый из которых реализует функцию F1, F2, F3 из лабораторной работы номер 1.

Требования к созданию потоков и необходимые исследования программы описаны в

лабораторной работе 2.

**Необходимые теоретические сведения:** язык Python обеспечивает возможность

программирования параллельных процессов. Теоретические сведения по программированию

потоков и управлению ими в языке Python можно найти на сайте

http://rus-linux.net/MyLDP/BOOKS/python.pdf

**Код программы :**  *""""   
Parallel programming  
Kobylynskiy Dmytro  
IP-31  
F1 = MIN(A + B) \* (B + C) \*(MA\*MD)  
F2 = ML + MK \*MO  
F3 = SORT(S + T)\*TRANS(MS\*MR)  
29.11.2015  
"""***from** multiprocessing **import** Process  
**import** threading  
**import** time  
N = 3000  
**def** InputVector():  
 vector = [1 **for** col **in** range(N)]  
 **return** vector  
  
**def** InputMatrix():  
 matrix = [[1 **for** row **in** range(N)] **for** col **in** range(N)]  
 **return** matrix  
  
**def** OutputVector(vector):  
 **if** N <= 5:  
 **print**(vector)  
  
**def** OutputMatrix(matrix):  
 **if** N <= 5:  
 **for** i **in** range(N):  
 **print**(matrix[i])  
  
**def** MultiplyMatrices(matrix1, matrix2):  
 result = [[0 **for** row **in** range(len(matrix1))] **for** col **in** range(len(matrix2[0]))]  
 **for** i **in** range(len(matrix1)):  
 **for** j **in** range(len(matrix2[0])):  
 **for** k **in** range(len(matrix2)):  
 result[i][j] += matrix1[i][k] \* matrix2[k][j]  
 **return** result  
**def** MultiplyVectorOnMatrix(vector, matrix):  
 result = InputVector()  
 **for** i **in** range(N):  
 result[i] = 0  
 **for** j **in** range(N):  
 result[i] = result[i] + vector[j] \* matrix[j][i]  
 **return** result  
  
**def** MultiplyVectors(vector1, vector2):  
 vector = InputVector()  
 result = 0  
 **for** i **in** range(N):  
 vector[i] = 0  
 vector[i] = vector1[i] \* vector2[i]  
 result = result + vector[i]  
 **return** result  
  
**def** SumVectors(vector1, vector2):  
 result = InputVector()  
 **for** i **in** range(N):  
 result[i] = vector1[i] + vector2[i]  
 **return** result  
  
**def** SumMatrix(matrix1, matrix2):  
 result = InputMatrix()  
 **for** i **in** range(N):  
 **for** j **in** range(N):  
 result[i][j] = matrix1[i][j] + matrix2[i][j]  
 **return** result  
  
**def** SortVector(vector):  
 **for** i **in** range(N):  
 **for** j **in** range(N):  
 **if** (vector[i] > vector[j]):  
 buffer = vector[i]  
 vector[i] = vector[j]  
 vector[j] = buffer  
 **return** vector  
  
**def** SortMatrix(matrix):  
 **for** i **in** range(N):  
 matrix[i] = SortVector(matrix[i])  
  
**def** MinVector(vector):  
 min = vector[0]  
 **for** i **in** range(N):  
 **if** (vector[i] < min):  
 vector = vector[i]  
 **return** min  
  
**def** MultiplyMatrixOnDigit(matrix, digit):  
 **for** i **in** range(N):  
 **for** j **in** range(N):  
 matrix[i][j] = digit \* matrix[i][j]  
 **return** matrix  
**def** MultiplyVectorOnDigit(vector, digit):  
 **for** i **in** range(N):  
 vector[i] = digit \* vector[i]  
 **return** vector  
**def** Transpose(matrix):  
 **for** i **in** range(N):  
 **for** j **in** range(N):  
 matrix[i][j] = matrix[j][i]  
 **return** matrix  
  
**def** F1():  
 time.sleep(0.1)  
 **print**(**"Task1 started."**)  
 A = InputVector()  
 B = InputVector()  
 C = InputVector()  
 MA = InputMatrix()  
 MD = InputMatrix()  
 RES = MultiplyMatrices(MA, MD)  
 R = SumVectors(B, C)  
 R1 = MultiplyVectorOnMatrix(R,RES)  
 A = SumVectors(A,B)  
 min = MinVector(A)  
 R2 = MultiplyVectorOnDigit(R1,min)  
 **if** N <= 5:  
 OutputVector(R2)  
 **print**(**"Task1 ended."**)  
  
**def** F2():  
 time.sleep(0.2)  
 **print**(**"Task2 started."**)  
 MK = InputMatrix()  
 ML = InputMatrix()  
 MO = InputMatrix()  
 MF = MultiplyMatrices(MK, MO)  
 RES = SumMatrix(MF,MK)  
 OutputMatrix(RES)  
 **print**(**"Task2 ended."**)  
  
**def** F3():  
 time.sleep(0.3)  
 **print**(**"Task3 started."**)  
 MS = InputMatrix()  
 MR = InputMatrix()  
 S = InputVector()  
 T = InputVector()  
 R1 = SumVectors(S,T)  
 R1 = SortVector(R1)  
 RM1 = MultiplyMatrices(MS,MR)  
 MF = Transpose(RM1)  
 RES = MultiplyVectorOnMatrix(R1,MF)  
 **if** N <= 5:  
 OutputMatrix(RES)  
 **print**(**"Task3 ended."**)  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **print**(**"Lab8."**)  
 process1 = Process(target=F1)  
 process2 = Process(target=F2)  
 process3 = Process(target=F3)  
 process1.start()  
 process2.start()  
 process3.start()  
  
 process1.join()  
 **print**(**"Lab8 ended."**)  
 input()