НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №7**

з дисципліни **«**Паралельні та розподілені обчислення**»**

Виконав:

студент 3 курсу гр. ІО-42

Кочетов Данило

№ ЗК 4213

Перевірив:

Долголенко О. М.

Київ 2016 р.

***Завдання:***

1.13; 2.13; 3.13

F1: C = A\*(MA\*ME) + B + D

F2: ML = MIN(MF)\*MG + MAX(MH) \* (MK\*MF)

F3: T = (MO\*MP)\*S + MR\*SORT(S)

***Лістинг програми:***

***# Lab7.py***

from threading import Thread

from F1 import F1

from F2 import F2

from F3 import F3

N = 100

print('Lab 7 start\n')

f1, f2, f3 = F1(N), F2(N), F3(N)

t1, t2, t3 = Thread(target = f1.run), Thread(target = f2.run), Thread(target = f3.run)

t1.start()

t2.start()

t3.start()

t1.join()

t2.join()

t3.join()

#print(f1.result)

#print(f2.result)

#print(f3.result)

print('\nLab 7 end\n')

print('Press Enter...')

input()

***# F1.py***

from Matrix import Matrix

from Vector import Vector

class F1:

def \_\_init\_\_(self, N):

self.N = N

def run(self):

print('Task 1 start')

MA, ME = [Matrix(self.N) for \_ in range(2)]

A, B, D = [Vector(self.N) for \_ in range(3)]

self.result = MA.multiply(ME).multiply(A).sum(B).sum(D)

print('Task 1 end')

***# F2.py***

from Matrix import Matrix

from Vector import Vector

class F2:

def \_\_init\_\_(self, N):

self.N = N

def run(self):

print('Task 2 start')

MG, MF, MK, MH = [Matrix(self.N) for \_ in range(4)]

self.result = MG.multiply(MF.min()).sum(MK.multiply(MF).multiply(MH.max()))

print('Task 2 end')

***# F3.py***

from Matrix import Matrix

from Vector import Vector

class F3:

def \_\_init\_\_(self, N):

self.N = N

def run(self):

print('Task 3 start')

MO, MP, MR = [Matrix(self.N) for \_ in range(3)]

S = Vector(self.N)

self.result = MO.multiply(MP).multiply(S).sum(MR.multiply(S.sort()))

print('Task 3 end')

***# Vector.py***

from random import randrange

class Vector:

def \_\_init\_\_(self, param):

if (type(param) is int):

self.N, self.grid = param, [randrange(10, 50) for \_ in range(param)]

else:

self.N, self.grid = len(param), param[:]

def \_\_repr\_\_(self):

return ' '.join(map(str, self.grid))

def get(self, i):

return self.grid[i]

def sum(self, v):

return Vector([self.grid[i] + v.get(i) for i in range(v.N)])

def sort(self):

return Vector(sorted(self.grid))

***# Matrix.py***

from random import randrange

from Vector import Vector

class Matrix:

def \_\_init\_\_(self, param):

if (type(param) is int):

self.N, self.grid = param, [[randrange(10, 50) for \_ in range(param)] for \_\_ in range(param)]

else:

self.N, self.grid = len(param), param[:]

def \_\_repr\_\_(self):

return ('\n'.join([' '.join([str(item) for item in row]) for row in self.grid]))

def get(self, i, k):

return self.grid[i][k]

def multiply(self, param):

if (type(param) is Matrix):

return Matrix([[sum([self.grid[i][j] \* param.get(j, k) for j in range(self.N)]) for k in range(self.N)] for i in range(self.N)])

elif (type(param) is Vector):

return Vector([sum([self.grid[i][j] \* param.get(j) for j in range(self.N)]) for i in range(self.N)])

else:

return Matrix([[item \* param for item in row] for row in self.grid])

def sum(self, m):

return Matrix([[self.grid[i][k] + m.get(i, k) for k in range(self.N)] for i in range(self.N)])

def min(self):

return min([min([item for item in row]) for row in self.grid])

def max(self):

return max([max([item for item in row]) for row in self.grid])