КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Кафедра інтелектуальних та інформаційних систем

Лабораторна робота № 2

з дисципліни

“Методи та системи паралельного програмування”

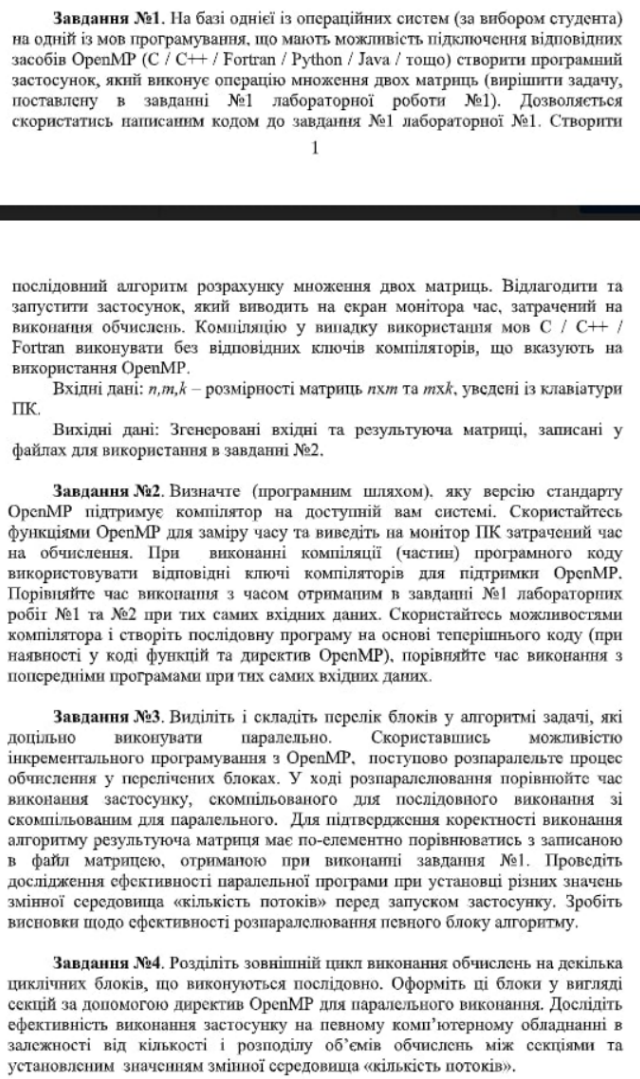
Виконав студент

групи КН- 31

Пашковський Павло Володимирович

Київ-2020

Завдання:



**1 Завдання**

from docutils.nodes import inline  
from numpy import \*  
from random import \*  
from time import time  
print()  
# для приятной печати времени  
MACRO = 100  
  
# openMP код  
codeOpenMP = \  
 """  
 int i = 0;  
   
 omp\_set\_num\_threads(2);  
 #pragma omp parallel shared(matrix, randRow, c) private(i) private(C)  
 {  
 #pragma omp for   
 for(i = 0; i < N\*M; i++) {  
 matrix[0,i] = matrix[0,i] - (c \* randRow[i%M]);  
 }  
 int C[N][M]=0;  
 #pragma omp for private(C)  
 for(i = 0; i < N; i++)  
 for(j = 0; j < M; j++)  
 {  
 C[i][j] = 0;  
 for(k = 0; k < N; k++)  
 C[i][j] += A[i][k] \* B[k][j];  
 }  
 }  
 """  
  
  
  
  
# генерация случайной матрицы x на y  
# элементы матрицы - случайные числа от 0 до 99 включ.  
def randMat(x, y):  
 randRaw = lambda a: [randint(0, 100) for i in range(0, a)]  
 randConst = lambda x, y: [randRaw(x) for e in range(0, y)]  
 return array(randConst(x, y))  
  
  
def test():  
 razmer = [100, 1000, 2000, 3000]  
  
 for n in razmer:  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
  
 print("\nTest on size: %dx%d" % (n, n))  
 import numpy as np  
 """ python test """  
 matrix = array(sourceMat)  
 t1 = time()  
 for i in range(N):  
 matrix[i, :] -= c \* randRow  
  
 m=np.matrix(matrix)  
 m=m\*m  
 timePython = (time() - t1) \* MACRO  
 print("\tPure python: ", timePython)  
  
  
  
 """ C и OpenMP test """  
 matrix = array(sourceMat)  
 t1 = time()  
 inline(codeOpenMP, ['matrix', 'c', 'randRow', 'N', 'M'],  
 extra\_compile\_args=['-O3 -fopenmp'],  
 compiler='gcc',  
 libraries=['gomp'],  
 headers=['<omp.h>'])  
 timeOpenMP = (time() - t1) \* MACRO  
 print("\tC plus OpenMP: %s" % (timeOpenMP))  
  
  
  
test()

Частина на Си:

"""  
 int i = 0;  
   
 omp\_set\_num\_threads(2);  
 #pragma omp parallel shared(matrix, randRow, c) private(i) private(C)  
 {  
 #pragma omp for   
 for(i = 0; i < N\*M; i++) {  
 matrix[0,i] = matrix[0,i] - (c \* randRow[i%M]);  
 }  
 int C[N][M]=0;  
 #pragma omp for private(C)  
 for(i = 0; i < N; i++)  
 for(j = 0; j < M; j++)  
 {  
 C[i][j] = 0;  
 for(k = 0; k < N; k++)  
 C[i][j] += A[i][k] \* B[k][j];  
 }  
 }  
 """

**Частина на Python:**

matrix = array(sourceMat)  
 t1 = time()  
 for i in range(N):  
 matrix[i, :] -= c \* randRow  
  
 m=np.matrix(matrix)  
 m=m\*m  
 timePython = (time() - t1) \* MACRO  
 print("\tPure python: ", timePython)

**2 Завдання**

Тести:

Test on size: 100x100

Pure python: 0.25818347930908203

C plus OpenMP: 0.0

Test on size: 1000x1000

Pure python: 106.6180944442749

C plus OpenMP: 0.0

Test on size: 2000x2000

Pure python: 3099.117684364319

C plus OpenMP: 0.09963512420654297

Test on size: 3000x3000

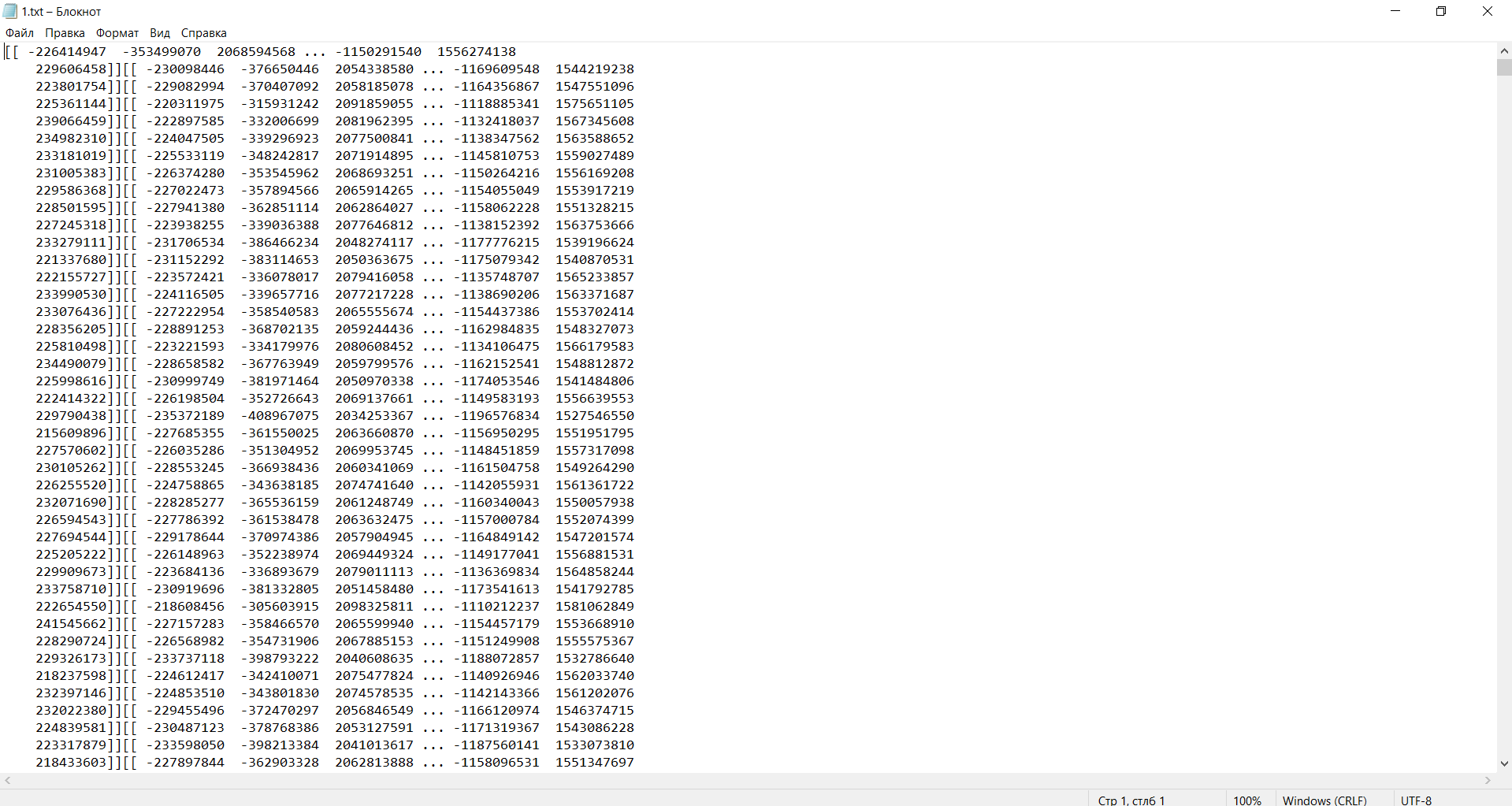
Функція для часу:

t1 = time()

timePython = (time() - t1) \* MACRO

**3 Завдання**

from threading import Thread  
  
from docutils.nodes import inline  
from numpy import \*  
from random import \*  
from time import time  
print()  
# для приятной печати времени  
MACRO = 100  
  
# openMP код  
codeOpenMP = \  
 """  
 int i = 0;  
   
 omp\_set\_num\_threads(2);  
 #pragma omp parallel shared(matrix, randRow, c) private(i) private(C)  
 {  
 #pragma omp for   
 for(i = 0; i < N\*M; i++) {  
 matrix[0,i] = matrix[0,i] - (c \* randRow[i%M]);  
 }  
 int C[N][M]=0;  
 #pragma omp for private(C)  
 for(i = 0; i < N; i++)  
 for(j = 0; j < M; j++)  
 {  
 C[i][j] = 0;  
 for(k = 0; k < N; k++)  
 C[i][j] += A[i][k] \* B[k][j];  
 }  
 }  
 """  
  
  
  
  
# генерация случайной матрицы x на y  
# элементы матрицы - случайные числа от 0 до 99 включ.  
def randMat(x, y):  
 randRaw = lambda a: [randint(0, 100) for i in range(0, a)]  
 randConst = lambda x, y: [randRaw(x) for e in range(0, y)]  
 return array(randConst(x, y))  
  
  
def test():  
 razmer = [100, 1000, 2000, 3000]  
 for n in razmer:  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
  
 #print("\nTest on size: %dx%d" % (n, n))  
 import numpy as np  
 """ python test """  
 matrix = array(sourceMat)  
 t1 = time()  
 for i in range(N):  
 matrix[i, :] -= c \* randRow  
  
 m=np.matrix(matrix)  
 m=m\*m  
 f=open('1.txt','w')  
 for i in range(len(m)):  
 g=str(m[i])  
 f.write(g)  
 timePython = (time() - t1) \* MACRO  
 print("\tPure python: ", timePython,'\n')  
  
def test1():  
 razmer = [100, 1000, 2000, 3000]  
 for n in razmer:  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
 #print("\nTest on size: %dx%d" % (n, n))  
 import numpy as np  
 """ C и OpenMP test """  
 matrix = array(sourceMat)  
 t1 = time()  
 inline(codeOpenMP, ['matrix', 'c', 'randRow', 'N', 'M'],  
 extra\_compile\_args=['-O3 -fopenmp'],  
 compiler='gcc',  
 libraries=['gomp'],  
 headers=['<omp.h>'])  
 timeOpenMP = (time() - t1) \* MACRO  
  
 print("\tC plus OpenMP: %s\n" % (timeOpenMP))  
  
  
thread1 = Thread(target=test)# экземпляр потока Thread  
thread1.start()# Чтобы запустить отдельный поток, нужно создать экземпляр потока Thread и затем запустить его с помощью метода .start()  
thread2=Thread(target=test1)# экземпляр потока Thread  
thread2.start()



**….**

Записуємо й провіряємо

f=open('1.txt','w')  
 for i in range(len(m)):  
 g=str(m[i])  
 f.write(g)

**4 Завдання**

from threading import Thread  
  
from docutils.nodes import inline  
from numpy import \*  
from random import \*  
from time import time  
print()  
# для приятной печати времени  
MACRO = 100  
  
# openMP код  
codeOpenMP = \  
 """  
 int i = 0;  
   
 omp\_set\_num\_threads(2);  
 #pragma omp parallel shared(matrix, randRow, c) private(i) private(C)  
 {  
 #pragma omp for   
 for(i = 0; i < N\*M; i++) {  
 matrix[0,i] = matrix[0,i] - (c \* randRow[i%M]);  
 }  
 int C[N][M]=0;  
 #pragma omp for private(C)  
 for(i = 0; i < N; i++)  
 for(j = 0; j < M; j++)  
 {  
 C[i][j] = 0;  
 for(k = 0; k < N; k++)  
 C[i][j] += A[i][k] \* B[k][j];  
 }  
 }  
 """  
  
  
  
  
# генерация случайной матрицы x на y  
# элементы матрицы - случайные числа от 0 до 99 включ.  
def randMat(x, y):  
 randRaw = lambda a: [randint(0, 100) for i in range(0, a)]  
 randConst = lambda x, y: [randRaw(x) for e in range(0, y)]  
 return array(randConst(x, y))  
  
  
def test():  
 razmer = [100, 1000, 2000, 3000]  
 for n in razmer:  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
  
 #print("\nTest on size: %dx%d" % (n, n))  
 import numpy as np  
 """ python test """  
 matrix = array(sourceMat)  
 t1 = time()  
 for i in range(N):  
 matrix[i, :] -= c \* randRow  
  
 m=np.matrix(matrix)  
 m=m\*m  
  
 timePython = (time() - t1) \* MACRO  
 print("\tPure python: ", timePython,'\n')  
  
def test1():  
 razmer = [100, 1000, 2000, 3000]  
 for n in razmer:  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
 #print("\nTest on size: %dx%d" % (n, n))  
 import numpy as np  
 """ C и OpenMP test """  
 matrix = array(sourceMat)  
 t1 = time()  
 inline(codeOpenMP, ['matrix', 'c', 'randRow', 'N', 'M'],  
 extra\_compile\_args=['-O3 -fopenmp'],  
 compiler='gcc',  
 libraries=['gomp'],  
 headers=['<omp.h>'])  
 timeOpenMP = (time() - t1) \* MACRO  
  
 print("\tC plus OpenMP: %s\n" % (timeOpenMP))  
  
  
thread1 = Thread(target=test)# экземпляр потока Thread  
thread1.start()# Чтобы запустить отдельный поток, нужно создать экземпляр потока Thread и затем запустить его с помощью метода .start()  
thread2=Thread(target=test1)# экземпляр потока Thread  
thread2.start()