КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Кафедра інтелектуальних та інформаційних систем

Лабораторна робота № 4

з дисципліни

“Методи та системи паралельного програмування”

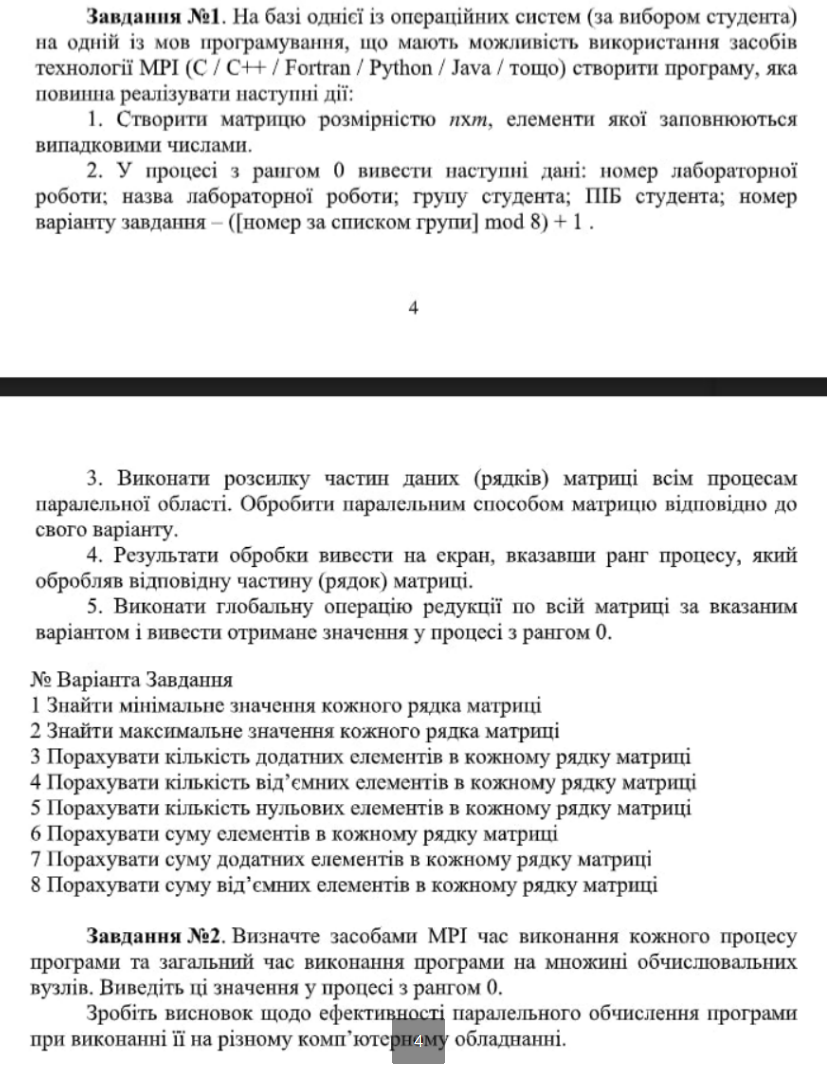
Виконав студент

групи КН-31

Пашковський Павло Володимирович

Київ-2020

Завдання:



**1 Завдання**

from threading import Thread  
  
import numpy  
from docutils.nodes import inline  
from numpy import \*  
from random import \*  
from time import time  
  
  
# 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 a():  
 # для приятной печати времени  
 MACRO = 50  
 # C и openMP код  
 codeOpenMP = \  
 """  
 int i = 0;  
  
 omp\_set\_num\_threads(2);  
 #pragma omp parallel shared(matrix, randRow, c) private(i)  
 {  
 #pragma omp for   
 int pk = 0;  
 int min = 0;  
 int nenul=0;  
 for(i = 0 ; i < N\*M - 1; i++) {  
 // сравниваем два соседних элемента.  
 for(j = 0 ; j < N\*M - i - 1 ; j++) {  
 if (matrix[j]>0){  
 pk=pk+1;  
 }  
 if (matrix[j]<0){  
 min=min+1;  
 }  
 if (matrix[j]!=0){  
 nenul=nenul+1;  
 }  
 if(matrix[j] > matrix[j+1]) {  
 // если они идут в неправильном порядке, то  
 // меняем их местами.  
 int tmp = matrix[j];  
 matrix[j] = matrix[j+1] ;  
 matrix[j+1] = tmp;  
 }  
 }  
 }  
 a1=matrix[-1]  
 b1=matrix[0]  
  
 }  
 """  
 results = []  
 n = 5000  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
 """ 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))  
 results.append(matrix)  
  
  
def b():  
 # для приятной печати времени  
 MACRO = 50  
 results = []  
 n = 50  
 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))  
  
 """ python test """  
 matrix = array(sourceMat)  
 t1 = time()  
 matrix.sort()  
 for i in range(len(matrix)):  
 min=matrix[i][0]  
 #print(f'Минимальный - {matrix[i][0]}')  
 for i in range(len(matrix)):  
 max=matrix[i][-1]  
 #print(f'Максимальный - {matrix[i][-1]}')  
 for i in range(len(matrix)):  
 sum1=sum(matrix[i])  
 doda=0  
 vid=0  
 nul=0  
 for i in range(len(matrix)):  
 for j in range(len(matrix[0])):  
 if matrix[i][j]>0:  
 doda+=1  
 elif matrix[i][j]<0:  
 vid+=1  
 else:  
 nul+=1  
  
 for i in range(len(matrix)):  
 synp = 0  
 symm = 0  
 for j in range(len(matrix[0])):  
 if matrix[i][j]>0:  
 synp+=matrix[i][j]  
 elif matrix[i][j]<0:  
 symm+=matrix[i][j]  
 timePython = (time() - t1) \* MACRO  
 print("\tPure python: ", timePython)  
 results.append(matrix)  
  
  
a()

b()

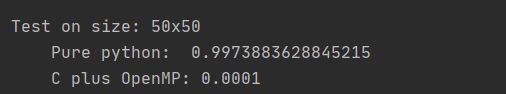
Частина на Си:

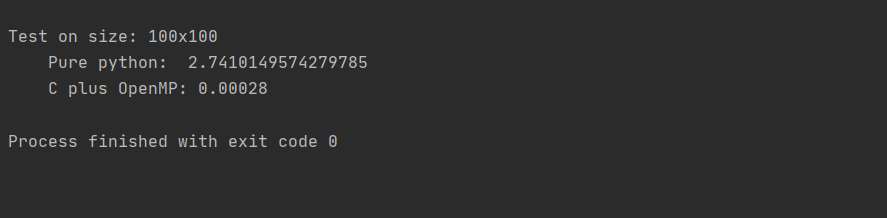
"""  
int i = 0;  
  
omp\_set\_num\_threads(2);  
#pragma omp parallel shared(matrix, randRow, c) private(i)  
{  
#pragma omp for   
int pk = 0;  
int min = 0;  
int nenul=0;  
for(i = 0 ; i < N\*M - 1; i++) {  
 // сравниваем два соседних элемента.  
 for(j = 0 ; j < N\*M - i - 1 ; j++) {  
 if (matrix[j]>0){  
 pk=pk+1;  
 }  
 if (matrix[j]<0){  
 min=min+1;  
 }  
 if (matrix[j]!=0){  
 nenul=nenul+1;  
 }  
 if(matrix[j] > matrix[j+1]) {  
 // если они идут в неправильном порядке, то  
 // меняем их местами.  
 int tmp = matrix[j];  
 matrix[j] = matrix[j+1] ;  
 matrix[j+1] = tmp;  
 }  
 }  
}  
a1=matrix[-1]  
b1=matrix[0]  
  
}  
"""

**Частина на Python:**

matrix = array(sourceMat)  
t1 = time()  
matrix.sort()  
for i in range(len(matrix)):  
 min=matrix[i][0]  
 #print(f'Минимальный - {matrix[i][0]}')  
for i in range(len(matrix)):  
 max=matrix[i][-1]  
 #print(f'Максимальный - {matrix[i][-1]}')  
for i in range(len(matrix)):  
 sum1=sum(matrix[i])  
doda=0  
vid=0  
nul=0  
for i in range(len(matrix)):  
 for j in range(len(matrix[0])):  
 if matrix[i][j]>0:  
 doda+=1  
 elif matrix[i][j]<0:  
 vid+=1  
 else:  
 nul+=1  
  
for i in range(len(matrix)):  
 synp = 0  
 symm = 0  
 for j in range(len(matrix[0])):  
 if matrix[i][j]>0:  
 synp+=matrix[i][j]  
 elif matrix[i][j]<0:  
 symm+=matrix[i][j]

Тест:





**2 Завдання**

**потоки**

thread1 = Thread(target=a)# 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=b)# экземпляр потока Thread  
thread2.start()# Чтобы запустить отдельный поток, нужно создать экземпляр потока Thread и затем запустить его с помощью метода .start()

**Функції часу**

timeOpenMP = (time() - t1) \* MACRO  
 print("\tC plus OpenMP: %s" % (timeOpenMP))

**Функції часу**

timeOpenMP = (time() - t1) \* MACRO  
print("\tC plus OpenMP: %s" % (timeOpenMP))

**Фунція визову си**

matrix = array(sourceMat)  
t1 = time()  
inline(codeOpenMP, ['matrix', 'c', 'randRow', 'N', 'M'],  
 extra\_compile\_args=['-O3 -fopenmp'],  
 compiler='gcc',  
 libraries=['gomp'],  
 headers=['<omp.h>'])