**ДОДАТКИ**

**Додаток А**

**Лістинг програми ПРГ1**

GNAT GPL 2014 (20140331)

Copyright 1992-2014, Free Software Foundation, Inc.

Compiling: C:\Users\Serhiy\Documents\adaprjs\Lab5\main.adb (source file time stamp: 2014-05-29 10:36:28)

1. with Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar;

2. use Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Calendar;

3.

4. procedure Main is

5. N: Integer := 1000;

6. P: Integer := 10;

7. H: Integer := N / P;

8. StartTime, FinishTime: Time;

9. DiffTime: Duration;

10.

11. type Vector is array(1..N) of Integer;

12. type Matrix is array(1..N) of Vector;

13.

14. --l: Integer;

15. MA, MO, ME, MK, MT, MM, Res: Matrix;

16.

17. procedure MatrixInput(M: out Matrix) is

18. begin

19. for i in 1..N loop

20. for j in 1..N loop

21. M(i)(j) := 1;

22. end loop;

23. end loop;

24. end MatrixInput;

25.

26. procedure MatrixOutput(M: in Matrix) is

27. begin

28. for i in 1..N loop

29. for j in 1..(N-1) loop

30. put(M(i)(j));

31. put(" ");

32. end loop;

33. put(M(i)(N));

34. Put\_Line("");

35. end loop;

36. end MatrixOutput;

37.

38. protected ResourceMonitor is

39. procedure WriteME(M: in Matrix);

40. procedure WriteMT(M: in Matrix);

41. procedure WriteL(lInp: in Integer);

42. procedure CalculateE(ei: in Integer);

43. function CopyE return Integer;

44. function CopyL return Integer;

45. function CopyME return Matrix;

46. function CopyMT return Matrix;

47. private

48. l: Integer;

49. e: Integer := Integer'Last;

50. ME: Matrix;

51. MT: Matrix;

52. end ResourceMonitor;

53.

54. protected SynchronizeMonitor is

55. procedure SignalInput;

56. procedure SignalCalculate2;

57. procedure SignalCalculate3;

58. entry WaitInput;

59. entry WaitCalculate2;

60. entry WaitCalculate3;

61. private

62. flagInput: Integer := 0;

63. flagCalculate2: Integer := 0;

64. flagCalculate3: Integer := 0;

65. end SynchronizeMonitor;

66.

67. protected body ResourceMonitor is

68. procedure WriteME(M: in Matrix) is

69. begin

70. ME := M;

71. end WriteME;

72.

73. procedure WriteMT(M: in Matrix) is

74. begin

75. MT := M;

76. end WriteMT;

77.

78. procedure WriteL(lInp: in Integer) is

79. begin

80. l := lInp;

81. end WriteL;

82.

83. procedure CalculateE(ei: in Integer) is

84. begin

85. if ( e > ei ) then

86. e := ei;

87. end if;

88. end CalculateE;

89.

90. function CopyE return Integer is

91. begin

92. return e;

93. end CopyE;

94.

95. function CopyL return Integer is

96. begin

97. return l;

98. end CopyL;

99.

100. function CopyME return Matrix is

101. begin

102. return ME;

103. end CopyME;

104.

105. function CopyMT return Matrix is

106. begin

107. return MT;

108. end CopyMT;

109. end ResourceMonitor;

110.

111. protected body SynchronizeMonitor is

112. procedure SignalInput is

113. begin

114. flagInput := flagInput + 1;

115. end SignalInput;

116.

117. procedure SignalCalculate2 is

118. begin

119. flagCalculate2 := flagCalculate2 + 1;

120. end SignalCalculate2;

121.

122. procedure SignalCalculate3 is

123. begin

124. flagCalculate3 := flagCalculate3 + 1;

125. end SignalCalculate3;

126.

127. entry WaitInput

128. when flagInput = 2 is

129. begin

130. null;

131. end WaitInput;

132.

133. entry WaitCalculate2

134. when flagCalculate2 = P is

|

>>> warning: potentially unsynchronized barrier

>>> warning: "P" should be private component of type

135. begin

136. null;

137. end WaitCalculate2;

138.

139. entry WaitCalculate3

140. when flagCalculate3 = P is

|

>>> warning: potentially unsynchronized barrier

>>> warning: "P" should be private component of type

141. begin

142. null;

143. end WaitCalculate3;

144.

145. end SynchronizeMonitor;

146.

147. task type CalculateTask(taskNumber: Integer);

148.

149. task body CalculateTask is

150. first, last: Integer;

151. sum: Integer;

152. ei: Integer;

153. li: Integer;

154. lInput : Integer;

155. MEi, MTi: Matrix;

156. begin

157. Put\_Line("Task " & Integer'Image(taskNumber) & " started...");

158.

159. if ( taskNumber = 1 ) then

160. -- ÐÐ²ÐµÐ´ÐµÐ½Ð½Ñ A, ME, MK

161. MatrixInput(MA);

162. MatrixInput(ME);

163. MatrixInput(MO);

164. ResourceMonitor.WriteME(ME);

165. -- Ð¡Ð¸Ð³Ð½Ð°Ð» T\_j (j = 2..P) Ð¿ÑÐ¾ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð²Ð²Ð¾Ð´Ñ

166. SynchronizeMonitor.SignalInput;

167. end if;

168.

169. if ( taskNumber = P ) then

170. -- ÐÐ²ÐµÐ´ÐµÐ½Ð½Ñ B, MB, MO

171. --l := 1;

172. MatrixInput(MK);

173. MatrixInput(MT);

174. MatrixInput(MM);

175. lInput := 1;

176. ResourceMonitor.WriteMT(MT);

177. ResourceMonitor.WriteL(lInput);

178.

179. -- Ð¡Ð¸Ð³Ð½Ð°Ð» T\_j (j = 1..P-1) Ð¿ÑÐ¾ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð²Ð²Ð¾Ð´Ñ

180. SynchronizeMonitor.SignalInput;

181. end if;

182.

183. -- Ð§ÐµÐºÐ°ÑÐ¸ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð²Ð²ÐµÐ´ÐµÐ½Ð½Ñ

184. SynchronizeMonitor.WaitInput;

185.

186. first := (taskNumber - 1) \* H + 1;

187. last := taskNumber \* H;

188.

189. -- ÐÐ±ÑÐ¸ÑÐ»ÐµÐ½Ð½Ñ a\_i

190. ei := Integer'Last;

191. for i in first..last loop

192. for j in 1..N loop

193. if ( ei > MO(i)(j) ) then

194. ei := ME(i)(j);

195. end if;

196. end loop;

197. end loop;

198.

199. -- ÐÐ±ÑÐ¸ÑÐ»ÐµÐ½Ð½Ñ a

200. ResourceMonitor.CalculateE(ei);

201.

202. SynchronizeMonitor.SignalCalculate2;

203.

204. SynchronizeMonitor.WaitCalculate2;

205.

206. li := ResourceMonitor.CopyL;

207. ei := ResourceMonitor.CopyE;

208. MEi := ResourceMonitor.CopyME;

209. MTi := ResourceMonitor.CopyMT;

210.

211. for i in first..last loop

212. for j in 1..N loop

213. sum := 0;

214. for k in 1..N loop

215. sum := sum + MT(j)(k)\*MM(k)(i);

216. end loop;

217. Res(j)(i) := sum;

218. end loop;

219. end loop;

220.

221. for i in first..last loop

222. for j in 1..N loop

223. sum := 0;

224. for k in 1..N loop

225. sum := sum + ME(j)(k)\*Res(k)(i);

226. end loop;

227. MA(j)(i) := ei\*MK(j)(i)+li\*sum;

228. end loop;

229. end loop;

230.

231. if ( taskNumber = 1 ) then

232. -- Ð§ÐµÐºÐ°ÑÐ¸ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð¾Ð±ÑÐ¸ÑÐ»ÐµÐ½Ð½Ñ MA\_H Ð² T\_j (j = 2..P)

233. SynchronizeMonitor.SignalCalculate3;

234. SynchronizeMonitor.WaitCalculate3;

235.

236. -- ÐÐ¸Ð²ÐµÐ´ÐµÐ½Ð½Ñ MA

237. if ( N <= 8 ) then

238. MatrixOutput(MA);

239. end if;

240.

241. FinishTime := Clock;

242. DiffTime := FinishTime - StartTime;

243.

244. Put("Time = ");

245. Put(Integer(DiffTime), 1);

246. Put\_Line("");

247. else

248. -- Ð¡Ð¸Ð³Ð½Ð°Ð» T\_1 Ð¿ÑÐ¾ Ð·Ð°ÐºÑÐ½ÑÐµÐ½Ð½Ñ Ð¾Ð±ÑÐ¸ÑÐ»ÐµÐ½Ð½Ñ MA\_H

249. SynchronizeMonitor.SignalCalculate3;

250. end if;

251.

252. Put\_Line("Task " & Integer'Image(taskNumber) & " finished");

253. end CalculateTask;

254.

255. ---------------------------------------------------------------

256. -- Ð¢Ð¸Ð¿ Ð²ÐºÐ°Ð·ÑÐ²Ð½Ð¸ÐºÐ° Ð½Ð° Ð·Ð°Ð´Ð°ÑÑ

257. type CalculateTaskPointer is access CalculateTask;

258.

259. -- ÐÐ°ÑÑÐ¸Ð² Ð²ÐºÐ°Ð·ÑÐ²Ð½Ð¸ÐºÑÐ² Ð½Ð° Ð·Ð°Ð´Ð°ÑÑ

260. type TasksArray is array(1..P) of CalculateTaskPointer;

261.

262. tArray: TasksArray;

263.

264. ch : Character;

265.

266.

267. begin

268.

269. Get(ch);

270. StartTime := Clock;

271.

272. for i in 1..P loop

273. tArray(i) := new CalculateTask(i);

274. end loop;

275. end Main;

275 lines: No errors

**Додаток Б**

**Лістинг програми ПРГ2**

Клас Matrix.h:

#pragma once

#include <iostream>

class Matrix {

private:

static const int FILLER = 1;

//int size;

int rows;

int columns;

public:

int \*matrix;

Matrix(int size);

Matrix(int rows, int columns);

~Matrix();

int getRows();

int getColumns();

void\* getPtrToArray();//--

int\* operator [] (int row);

void input();

void copy(Matrix& copyMatrix);

void mult(Matrix& res, Matrix& multMatr);

int getMin();

int getMin(int from, int to);

void delExcessive(int from, int to);

void merge(Matrix& MAh);

};

std::ostream& operator << (std::ostream &out, Matrix &matrix);

Клас Matrix.cpp:

#include "Matrix.h"

#include <iostream>

Matrix::Matrix(int size) {

this->rows = size;

this->columns = size;

matrix = new int [size\*size];

}

Matrix::Matrix(int rows, int columns) {

this->rows = rows;

this->columns = columns;

matrix = new int [rows\*columns];

}

Matrix::~Matrix() {

delete(matrix);

}

int Matrix::getRows() {

return rows;

}

int Matrix::getColumns() {

return columns;

}

void\* Matrix::getPtrToArray(){

return (void\*)(&(\*matrix));

}

int\* Matrix::operator[](int row) {

return matrix + row\*columns;

}

void Matrix::input() {

for ( int i = 0; i < rows\*columns; i++ ) {

matrix[i] = FILLER;

}

}

void Matrix::copy(Matrix& copyMatrix) {

for ( int i = 0; i < rows; i++ ) {

for ( int j = 0; j < columns; j++ ) {

copyMatrix[i][j] = matrix[i\*columns + j];

}

}

}

void Matrix::mult(Matrix& res, Matrix& multMatr){

int\* r;

int\* mm;

int\* tm;

int rColumns = res.getColumns();

int mmColumns = multMatr.getColumns();

int tmColumns = columns;

for (int i = 0; i < mmColumns; i++)

{

r = res.matrix;

tm = matrix;

for (int j = 0; j < rows; j++)

{

r[i] = 0;

mm = multMatr.matrix;

for (int k = 0; k < rows; k++)

{

r[i] += tm[k] \* mm[i];

mm += mmColumns;

};

r += rColumns;

tm += tmColumns;

}

}

}

int Matrix::getMin(){

int min = matrix[0];

for(int i = 1; i < rows\*columns; i++){

if(min > matrix[i]){

min = matrix[i];

}

}

return min;

}

int Matrix::getMin(int from, int to){

int min = matrix[from];

for(int i = 0; i < rows; i++){

for(int j = from; j < to; j++){

if(min > matrix[i]){

min = matrix[i];

}

}

}

return min;

}

void Matrix::delExcessive(int from, int to){

int\* accum = new int[(to-from)\*rows];

for(int i = 0; i < rows; i++){

for(int j = from; j < to; j++){

accum[i\*(to-from) - from + j] = matrix[i\*columns + j];

}

}

int\* toDel = matrix;

matrix = accum;

delete[] toDel;

columns = to-from;

}

void Matrix::merge(Matrix& MAh){

int size1 = columns\*rows;

int size2 = MAh.getColumns()\*MAh.getRows();

int\* accum = new int[size1 + size2];

for(int i = 0; i < rows; i++){

for(int j = 0; j < (columns + MAh.getColumns()); j++){

if(j < columns){

accum[i\*(columns + MAh.getColumns()) + j] = matrix[i\*columns + j];

} else {

accum[i\*(columns + MAh.getColumns()) + j] = MAh[i][j-columns];

}

}

}

int\* toDel = matrix;

matrix = accum;

delete[] toDel;

columns += MAh.getColumns();

}

std::ostream& operator<<(std::ostream &out, Matrix &matrix) {

int last = matrix.getColumns() - 1;

out << "Matrix:" << std::endl;

for (int row = 0; row < matrix.getRows(); row++) {

for (int col = 0; col < last; col++)

out << matrix[row][col] << " ";

out << matrix[row][last] << std::endl;

}

return out;

};

Клас CourseWork(mpi).cpp:

//--------------------------

//Курсова робота

//MA = min(MО)\*MК + l\*ME\*(MT\*MM)

//Бута С.О.

//20.05.14

//--------------------------

#include "stdafx.h"

#include <mpi.h>

#include <iostream>

#include "Matrix.h"

#include <time.h>

using namespace std;

int N = 1000;

int P = 10;

int H = N/P;

int FILLER = 1;

int first = 0;

int last = P/2 - 1;

void\* shiftPtrLeft(void\* inp, int s){

return (void\*)((int\*)inp + s\*H\*N);

}

void copyBuf(void\* from, void\* to, int size){

for(int i = 0; i < size; i++){

((int\*)to)[i] = ((int\*)from)[i];

}

}

bool isCentral(int rank){

if(rank == (P/4))

return true;

return false;

}

int findAndSendMinGlobal(int rank, int localMin){

int ei;

int bottomRank = rank + P/2;

MPI\_Status st;

if(rank != last){

MPI\_Recv(&ei, 1, MPI\_INT, rank+1, 21, MPI\_COMM\_WORLD, &st);

localMin = (ei < localMin) ? ei : localMin;

}

MPI\_Recv(&ei, 1, MPI\_INT, rank-1, 21, MPI\_COMM\_WORLD, &st);

localMin = (ei < localMin) ? ei : localMin;

MPI\_Recv(&ei, 1, MPI\_INT, bottomRank, 21, MPI\_COMM\_WORLD, &st);

localMin = (ei < localMin) ? ei : localMin;

MPI\_Request rqb, rql;

MPI\_Isend(&localMin, 1, MPI\_INT, bottomRank, 21, MPI\_COMM\_WORLD, &rqb);

MPI\_Isend(&localMin, 1, MPI\_INT, rank-1, 21, MPI\_COMM\_WORLD, &rql);

if(rank != last){

MPI\_Request rqr;

MPI\_Isend(&localMin, 1, MPI\_INT, rank+1, 21, MPI\_COMM\_WORLD, &rqr);

MPI\_Wait(&rqr, MPI\_STATUS\_IGNORE);

}

MPI\_Wait(&rqb, MPI\_STATUS\_IGNORE);

MPI\_Wait(&rql, MPI\_STATUS\_IGNORE);

return localMin;

}

int getMinGlobal(int rank, int localMin){

int globalMin;

if(isCentral(rank)){

globalMin = findAndSendMinGlobal(rank, localMin);

} else {

int ei;

MPI\_Status st;

if(rank < P/4){

MPI\_Recv(&ei, 1, MPI\_INT, rank + P/2, 21, MPI\_COMM\_WORLD, &st);

localMin = (ei < localMin) ? ei : localMin;

if(rank != first){

MPI\_Recv(&ei, 1, MPI\_INT, rank - 1, 21, MPI\_COMM\_WORLD, &st);

localMin = (ei < localMin) ? ei : localMin;

}

MPI\_Send(&localMin, 1, MPI\_INT, rank+1, 21, MPI\_COMM\_WORLD);

MPI\_Recv(&globalMin, 1, MPI\_INT, rank+1, 21, MPI\_COMM\_WORLD, &st);

if(rank != first){

MPI\_Send(&globalMin, 1, MPI\_INT, rank-1, 21, MPI\_COMM\_WORLD);

}

MPI\_Send(&globalMin, 1, MPI\_INT, rank+P/2, 21, MPI\_COMM\_WORLD);

} else {

MPI\_Recv(&ei, 1, MPI\_INT, rank + P/2, 21, MPI\_COMM\_WORLD, &st);

localMin = (ei < localMin) ? ei : localMin;

if(rank != last){

MPI\_Recv(&ei, 1, MPI\_INT, rank + 1, 21, MPI\_COMM\_WORLD, &st);

localMin = (ei < localMin) ? ei : localMin;

}

MPI\_Send(&localMin, 1, MPI\_INT, rank-1, 21, MPI\_COMM\_WORLD);

MPI\_Recv(&globalMin, 1, MPI\_INT, rank-1, 21, MPI\_COMM\_WORLD, &st);

if(rank != last){

MPI\_Send(&globalMin, 1, MPI\_INT, rank+1, 21, MPI\_COMM\_WORLD);

}

MPI\_Send(&globalMin, 1, MPI\_INT, rank+P/2, 21, MPI\_COMM\_WORLD);

}

}

return globalMin;

}

void calcMAh(Matrix& MA, int e, Matrix& MKh, int l, Matrix& ME, Matrix& MT, Matrix& MMh){

Matrix Res(N, H);

MT.mult(Res, MMh);

ME.mult(MA, Res);

for(int i = 0; i < H; i++){

for(int j = 0; j < N; j++){

MA[j][i] = e\*MKh[j][i] + l\*MA[j][i];

}

}

}

int size(Matrix& MA){

return MA.getColumns()\*MA.getRows();

}

void matrOut(Matrix& M){

for(int i = 0; i < M.getRows(); i++){

for(int j = 0; j < M.getColumns(); j++){

cout << M[i][j] << " ";

}

cout << endl;

}

}

void sendLeftMAh(int rank, Matrix& MAh){

Matrix MAh2(N, H);

MPI\_Recv(MAh2.getPtrToArray(), N\*H, MPI\_INT, rank + P/2, 31, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MAh.merge(MAh2);

if(rank != last){

Matrix MAh1(N, H\*(P - (rank+1)\*2));

MPI\_Recv(MAh1.getPtrToArray(), size(MAh1), MPI\_INT, rank+1, 31, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MAh.merge(MAh1);

}

MPI\_Send(MAh.getPtrToArray(), size(MAh), MPI\_INT, rank-1, 31, MPI\_COMM\_WORLD);

}

void threadFuncFirst(){

long transferStart1 = clock();

Matrix MO(N), ME(N);

MO.input();

ME.input();

int rightSS = N\*N - 2\*H\*N;

int rightRS = 2\*N\*H;

int fullSize = N\*N;

int bottomRank = 0 + P/2;

int flag = -1;

MPI\_Request req0;

MPI\_Isend(&flag, 1, MPI\_INT, 1, 0, MPI\_COMM\_WORLD, &req0);

MPI\_Request req4;

MPI\_Request req5;

MPI\_Isend(shiftPtrLeft(MO.getPtrToArray(), 2), rightSS, MPI\_INT, 1, 4, MPI\_COMM\_WORLD, &req4);

MPI\_Isend(ME.getPtrToArray(), fullSize, MPI\_INT, 1, 5, MPI\_COMM\_WORLD, &req5);

Matrix MKh(N,2\*H), MMh(N,2\*H);

Matrix MT(N);

int li;

MPI\_Status st1;

MPI\_Status st2;

MPI\_Status st3;

MPI\_Recv(&flag, 1, MPI\_INT, 1, 0, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(MKh.getPtrToArray(), rightRS, MPI\_INT, 1, 1, MPI\_COMM\_WORLD, &st1);

MPI\_Recv(MMh.getPtrToArray(), rightRS, MPI\_INT, 1, 2, MPI\_COMM\_WORLD, &st2);

MPI\_Recv(MT.getPtrToArray(), fullSize, MPI\_INT, 1, 3, MPI\_COMM\_WORLD, &st3);

MPI\_Recv(&li, 1, MPI\_INT, 1, 6, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Wait(&req4, MPI\_STATUS\_IGNORE);

MPI\_Wait(&req5, MPI\_STATUS\_IGNORE);

MPI\_Request reql1, reql2, reql3, reql4, reql5, reql6;

MPI\_Isend(shiftPtrLeft(MKh.getPtrToArray(), 1), N\*H, MPI\_INT, bottomRank, 11, MPI\_COMM\_WORLD, &reql1);

MPI\_Isend(shiftPtrLeft(MMh.getPtrToArray(), 1), N\*H, MPI\_INT, bottomRank, 12, MPI\_COMM\_WORLD, &reql2);

MPI\_Isend(shiftPtrLeft(MO.getPtrToArray(), 1), N\*H, MPI\_INT, bottomRank, 14, MPI\_COMM\_WORLD, &reql4);

MPI\_Isend(MT.getPtrToArray(), N\*N, MPI\_INT, bottomRank, 13, MPI\_COMM\_WORLD, &reql3);

MPI\_Isend(ME.getPtrToArray(), N\*N, MPI\_INT, bottomRank, 15, MPI\_COMM\_WORLD, &reql5);

MPI\_Isend(&li, 1, MPI\_INT, bottomRank, 16, MPI\_COMM\_WORLD, &reql6);

MPI\_Wait(&reql1, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql2, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql3, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql4, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql5, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql6, MPI\_STATUS\_IGNORE);

MO.delExcessive(0, H);

MKh.delExcessive(0, H);

MMh.delExcessive(0, H);

long transferFinish1 = clock();

long cStart1 = clock();

int ei = MO.getMin();

long cFinish1 = clock();

int e = getMinGlobal(0, ei);

Matrix MAh(N, H);

long cStart2 = clock();

calcMAh(MAh, e, MKh, li, ME, MT, MMh);

long cFinish2 = clock();

long transferStart2 = clock();

Matrix MAh1(N, H);

MPI\_Recv(MAh1.getPtrToArray(), N\*H, MPI\_INT, bottomRank, 31, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MAh.merge(MAh1);

Matrix MAh2(N, N-2\*H);

MPI\_Recv(MAh2.getPtrToArray(), N\*(N-2\*H), MPI\_INT, 1, 31, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MAh.merge(MAh2);

long transferFinish2 = clock();

cout <<"Time to transfer: ";

cout << ((transferFinish1-transferStart1) + (transferFinish2 - transferStart2))/1000 << endl;

cout <<"Time to calculate1: ";

cout << (cFinish1 - cStart1)/1000 << endl;

cout <<"Time to calculate2: ";

cout << (cFinish2 - cStart2)/1000 << endl;

if(N < 13){

cout << "Matrix MA: \n";

for(int i = 0; i < MAh.getRows(); i++){

for(int j = 0; j < MAh.getColumns(); j++){

cout << MAh[i][j] << " ";

}

cout << endl;

}

}

}

void threadFuncLast(){

int bottomRank = P-1;

Matrix MK(N), MT(N), MM(N);

MK.input();

MT.input();

MM.input();

int li = FILLER;

int flag = 1;

MPI\_Request req0;

MPI\_Isend(&flag, 1, MPI\_INT, P/2-2, 0, MPI\_COMM\_WORLD, &req0);

MPI\_Request req1;

MPI\_Request req2;

MPI\_Request req3;

MPI\_Request req6;

MPI\_Isend(MK.getPtrToArray(), N\*N - 2\*H\*N, MPI\_INT, P/2-2, 1, MPI\_COMM\_WORLD, &req1);

MPI\_Isend(MM.getPtrToArray(), N\*N - 2\*H\*N, MPI\_INT, P/2-2, 2, MPI\_COMM\_WORLD, &req2);

MPI\_Isend(MT.getPtrToArray(), N\*N, MPI\_INT, P/2-2, 3, MPI\_COMM\_WORLD, &req3);

MPI\_Isend(&li, 1, MPI\_INT, P/2-2, 6, MPI\_COMM\_WORLD, &req6);

Matrix MOh(N, 2\*H), ME(N);

MPI\_Status st4;

MPI\_Status st5;

MPI\_Recv(&flag, 1, MPI\_INT, P/2-2, 0, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(MOh.getPtrToArray(), 2\*N\*H, MPI\_INT, P/2-2, 4, MPI\_COMM\_WORLD, &st4);

MPI\_Recv(ME.getPtrToArray(), N\*N, MPI\_INT, P/2-2, 5, MPI\_COMM\_WORLD, &st5);

MPI\_Wait(&req1, MPI\_STATUS\_IGNORE);

MPI\_Wait(&req2, MPI\_STATUS\_IGNORE);

MPI\_Wait(&req3, MPI\_STATUS\_IGNORE);

MPI\_Wait(&req6, MPI\_STATUS\_IGNORE);

MPI\_Request reql1, reql2, reql3, reql4, reql5/\*, reql6\*/;

MPI\_Isend(shiftPtrLeft(MK.getPtrToArray(), P - 1 ), N\*H, MPI\_INT, bottomRank, 11, MPI\_COMM\_WORLD, &reql1);

MPI\_Isend(shiftPtrLeft(MM.getPtrToArray(), 1), N\*H, MPI\_INT, bottomRank, 12, MPI\_COMM\_WORLD, &reql2);

MPI\_Isend(shiftPtrLeft(MOh.getPtrToArray(), 1), N\*H, MPI\_INT, bottomRank, 14, MPI\_COMM\_WORLD, &reql4);

MPI\_Isend(MT.getPtrToArray(), N\*N, MPI\_INT, bottomRank, 13, MPI\_COMM\_WORLD, &reql3);

MPI\_Isend(ME.getPtrToArray(), N\*N, MPI\_INT, bottomRank, 15, MPI\_COMM\_WORLD, &reql5);

MPI\_Isend(&li, 1, MPI\_INT, bottomRank, 16, MPI\_COMM\_WORLD, &reql5);

MPI\_Wait(&reql1, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql2, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql3, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql4, MPI\_STATUS\_IGNORE);

MPI\_Wait(&reql5, MPI\_STATUS\_IGNORE);

MK.delExcessive((P-2)\*H, (P-1)\*H);

MM.delExcessive((P-2)\*H, (P-1)\*H);

MOh.delExcessive(0, H);

int ei = MOh.getMin(0, H);

int e = getMinGlobal(last, ei);

Matrix MAh(N, H);

calcMAh(MAh, e, MK, li, ME, MT, MM);

sendLeftMAh(last, MAh);

}

void threadFuncMed(int rank){

int bottomRank = rank + P/2;

int rightSS = (P - rank\*2 - 2)\*H\*N;

int rightSR = (rank\*2 + 2)\*H\*N;

int leftSS = rank\*H\*2\*N;

int leftSR = (P-2\*rank)\*H\*N;

int flag;

MPI\_Request rq[7];

MPI\_Request rqf;

MPI\_Recv(&flag, 1, MPI\_INT, MPI\_ANY\_SOURCE, 0, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

Matrix MKh(N, rightSR/N), MMh(N, rightSR/N), MT(N);

int li;

Matrix MOh(N, leftSR/N), ME(N);

if(flag == 1){

MPI\_Recv(MKh.getPtrToArray(), rightSR, MPI\_INT, rank+1, 1, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(MMh.getPtrToArray(), rightSR, MPI\_INT, rank+1, 2, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(MT.getPtrToArray(), size(MT), MPI\_INT, rank+1, 3, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(&li, 1, MPI\_INT, rank+1, 6, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Isend(&flag, 1, MPI\_INT, rank-1, 0, MPI\_COMM\_WORLD, &rq[0]);

MPI\_Isend(MKh.getPtrToArray(), leftSS, MPI\_INT, rank-1, 1, MPI\_COMM\_WORLD, &rq[1]);

MPI\_Isend(MMh.getPtrToArray(), leftSS, MPI\_INT, rank-1, 2, MPI\_COMM\_WORLD, &rq[2]);

MPI\_Isend(MT.getPtrToArray(), size(MT), MPI\_INT, rank-1, 3, MPI\_COMM\_WORLD, &rq[3]);

MPI\_Isend(&li, 1, MPI\_INT, rank-1, 6, MPI\_COMM\_WORLD, &rq[6]);

MPI\_Recv(&flag, 1, MPI\_INT, MPI\_ANY\_SOURCE, 0, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(MOh.getPtrToArray(), leftSR, MPI\_INT, rank-1, 4, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(ME.getPtrToArray(), size(ME), MPI\_INT, rank-1, 5, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Isend(&flag, 1, MPI\_INT, rank+1, 0, MPI\_COMM\_WORLD, &rqf);

MPI\_Isend(shiftPtrLeft(MOh.getPtrToArray(), 2), rightSS, MPI\_INT, rank+1, 4, MPI\_COMM\_WORLD, &rq[4]);

MPI\_Isend(ME.getPtrToArray(), size(ME), MPI\_INT, rank+1, 5, MPI\_COMM\_WORLD, &rq[5]);

} else {

MPI\_Recv(MOh.getPtrToArray(), leftSR, MPI\_INT, rank-1, 4, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(ME.getPtrToArray(), size(ME), MPI\_INT, rank-1, 5, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Isend(&flag, 1, MPI\_INT, rank+1, 0, MPI\_COMM\_WORLD, &rqf);

MPI\_Isend(shiftPtrLeft(MOh.getPtrToArray(), 2), rightSS, MPI\_INT, rank+1, 4, MPI\_COMM\_WORLD, &rq[4]);

MPI\_Isend(ME.getPtrToArray(), size(ME), MPI\_INT, rank+1, 5, MPI\_COMM\_WORLD, &rq[5]);

MPI\_Recv(&flag, 1, MPI\_INT, MPI\_ANY\_SOURCE, 0, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(MKh.getPtrToArray(), rightSR, MPI\_INT, rank+1, 1, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(MMh.getPtrToArray(), rightSR, MPI\_INT, rank+1, 2, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(MT.getPtrToArray(), size(MT), MPI\_INT, rank+1, 3, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Recv(&li, 1, MPI\_INT, rank+1, 6, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

MPI\_Isend(&flag, 1, MPI\_INT, rank-1, 0, MPI\_COMM\_WORLD, &rq[0]);

MPI\_Isend(MKh.getPtrToArray(), leftSS, MPI\_INT, rank-1, 1, MPI\_COMM\_WORLD, &rq[1]);

MPI\_Isend(MMh.getPtrToArray(), leftSS, MPI\_INT, rank-1, 2, MPI\_COMM\_WORLD, &rq[2]);

MPI\_Isend(MT.getPtrToArray(), size(MT), MPI\_INT, rank-1, 3, MPI\_COMM\_WORLD, &rq[3]);

MPI\_Isend(&li, 1, MPI\_INT, rank-1, 6, MPI\_COMM\_WORLD, &rq[6]);

}

for(int i = 0; i < 7; i++){

MPI\_Wait(&rq[i], MPI\_STATUSES\_IGNORE);

}

MPI\_Wait(&rqf, MPI\_STATUSES\_IGNORE);

MKh.delExcessive(size(MKh)/N-2\*H, size(MKh)/N);

MMh.delExcessive(size(MMh)/N-2\*H, size(MMh)/N);

MOh.delExcessive(0, 2\*H);

MPI\_Request rq1[6];

MPI\_Isend(shiftPtrLeft(MKh.getPtrToArray(), 1), N\*H, MPI\_INT, bottomRank, 11, MPI\_COMM\_WORLD, &rq1[0]);

MPI\_Isend(shiftPtrLeft(MMh.getPtrToArray(), 1), size(MMh) - N\*H, MPI\_INT, bottomRank, 12, MPI\_COMM\_WORLD, &rq1[1]);

MPI\_Isend(MT.getPtrToArray(), size(MT), MPI\_INT, bottomRank, 13, MPI\_COMM\_WORLD, &rq1[2]);

MPI\_Isend(shiftPtrLeft(MOh.getPtrToArray(), 1), size(MOh) - N\*H, MPI\_INT, bottomRank, 14, MPI\_COMM\_WORLD, &rq1[3]);

MPI\_Isend(ME.getPtrToArray(), size(ME), MPI\_INT, bottomRank, 15, MPI\_COMM\_WORLD, &rq1[4]);

MPI\_Isend(&li, 1, MPI\_INT, bottomRank, 16, MPI\_COMM\_WORLD, &rq1[5]);

for(int i = 0; i < 6; i++){

MPI\_Wait(&rq1[i], MPI\_STATUSES\_IGNORE);

}

MKh.delExcessive(0, H);

MMh.delExcessive(0, H);

MOh.delExcessive(0, H);

int ei = MOh.getMin(0, H);

int e = getMinGlobal(rank, ei);

Matrix MAh(N, H);

calcMAh(MAh, e, MKh, li, ME, MT, MMh);

sendLeftMAh(rank, MAh);

}

void threadFuncBottom(int rank){

Matrix MOh(N, H), MKh(N, H), MMh(N, H), MT(N), ME(N);

int li;

MPI\_Status st1;

MPI\_Status st2;

MPI\_Status st3;

MPI\_Status st4;

MPI\_Status st5;

MPI\_Status st6;

MPI\_Recv(MKh.getPtrToArray(), N\*H, MPI\_INT, rank - P/2, 11, MPI\_COMM\_WORLD, &st1);

MPI\_Recv(MMh.getPtrToArray(), N\*H, MPI\_INT, rank - P/2, 12, MPI\_COMM\_WORLD, &st2);

MPI\_Recv(MT.getPtrToArray(), N\*N, MPI\_INT, rank - P/2, 13, MPI\_COMM\_WORLD, &st3);

MPI\_Recv(MOh.getPtrToArray(), N\*H, MPI\_INT, rank - P/2, 14, MPI\_COMM\_WORLD, &st4);

MPI\_Recv(ME.getPtrToArray(), N\*N, MPI\_INT, rank - P/2, 15, MPI\_COMM\_WORLD, &st5);

MPI\_Recv(&li, 1, MPI\_INT, rank - P/2, 16, MPI\_COMM\_WORLD, &st6);

int ei = MOh.getMin();

MPI\_Send(&ei, 1, MPI\_INT, rank - P/2, 21, MPI\_COMM\_WORLD);

int e;

MPI\_Recv(&e, 1, MPI\_INT, rank - P/2, 21, MPI\_COMM\_WORLD, MPI\_STATUSES\_IGNORE);

Matrix MAh(N, H);

calcMAh(MAh, e, MKh, li, ME, MT, MMh);

MPI\_Send(MAh.getPtrToArray(), N\*H, MPI\_INT, rank - P/2, 31, MPI\_COMM\_WORLD);

}

int main(int argc, char\* argv[])

{

MPI\_Init(&argc, &argv);

long tStart = clock();

int rank;

MPI\_Comm\_size(MPI\_COMM\_WORLD, &P);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

H = N/P;

last = P/2 - 1;

cout << "Thread " << rank << " started.\n";

if(rank == first){

threadFuncFirst();

} else {

if(rank == last){

threadFuncLast();

} else {

if(rank > (last)){

threadFuncBottom(rank);

} else {

threadFuncMed(rank);

}

}

}

cout << "Thread " << rank << " finish.\n";

MPI\_Finalize();

if(rank == first){

long tFinish = clock();

cout << endl << "time: " << (tFinish - tStart)/1000 << "s " << (tFinish - tStart)%1000 << "ms" << endl;

char ch;

cin >> ch;

}

return 0;

}

**Додаток В**



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

1

*Схема алгоритму процесів для ПКС із СП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

2

*Схема алгоритму процесів для ПКС із СП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

3

*Схема алгоритму процесу T1 для ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

4

*Схема алгоритму процесу TQ для ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

5

*Схема алгоритму процесу TQ для ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

6

*Схема алгоритму процесу TQ для ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

7

*Схема алгоритму процесу TQ для ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

8

*Схема алгоритму процесу TQ для ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

9

*Схема алгоритму процесу TQ для ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток В*

*Алгоритми процесів*

*Лит.*

*Листов*

9

*НТУУ “КПІ” ФІОТ*

**Додаток Г**

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

1

*Алгоритм основної програми для ПКС із СП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток Г*

*Алгоритми процесів*

*Лит.*

*Листов*

2

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

2

*Алгоритм основної програми для ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток Г*

*Алгоритми процесів*

*Лит.*

*Листов*

2

*НТУУ “КПІ” ФІОТ*



**Додаток Д**

*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

1

*Структурна схема ПКС із СП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток Д*

*Алгоритми процесів*

*Лит.*

*Листов*

2

*НТУУ “КПІ” ФІОТ*



*Изм.*

*Лист*

*№ докум.*

*Подпись*

*Дата*

*Лист*

2

*Структурна схема ПКС з ЛП*

*Розроб.*

*Бута С.О.*

*Провір.*

*Корочкін О.В.*

*Реценз.*

*Н. Контр.*

*Утверд.*

*Додаток Д*

*Алгоритми процесів*

*Лит.*

*Листов*

2

*НТУУ “КПІ” ФІОТ*

