НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА АВТОМАТИЗАЦІЇ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА УПРАВЛІННЯ

Лабораторна робота №7

з дисципліни **«**Паралельне програмування**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІП-31

Кахерський О.І.

Перевірив:

Корочкін О. В.

Київ – 2016 р.

**Тема:** Програмування для комп’ютерних систем зі локальною пам’яттю. Ада. Механізм рандеву

**Технічне завдання:** Розробити програму для розв’язання ПКС із ЛП (структура на рис. 1) математичної задачі: A =

**Мова програмування: Ада. Рандеву**

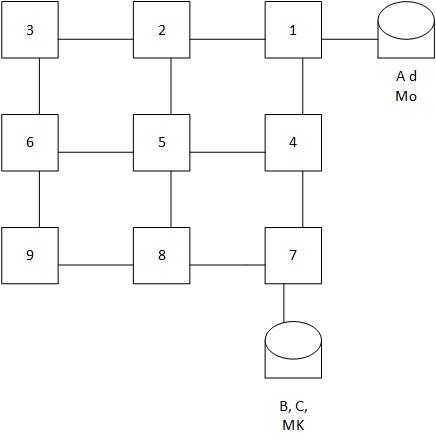


Рис.1 Структурна схема ПКС

**Виконання роботи**

**Етап 1.** Побудова паралельного алгоритму



**Етап 2. Розробка алогоритмів роботи кожного процесу**

**Задача Т1**

1. Ввід d, MO;
2. надіслати d, MO T4;
3. прийняти від Т4;
4. надіслати Т2;
5. обчислити
6. прийняти від Т2;
7. прийняти від Т4;
8. Вивід А.

**Задача Т2**

1. Прийняти від Т1;
2. Надіслати Т3;
3. обчислити
4. прийняти від Т3;
5. надіслати Т1.

**Задача Т3**

1. Прийняти від Т2;
2. обчислити
3. надіслати Т2;

**Задача Т4**

1. прийняти d, MO від Т1;
2. прийняти від Т7
3. надіслати d, MO Т7
4. надіслати Т1
5. надіслати Т5
6. обчислити
7. прийняти від Т5
8. прийняти від Т7
9. надіслати Т1

**Задача Т5**

1. Прийняти від Т4;
2. Надіслати Т6;
3. обчислити
4. прийняти від Т6;
5. надіслати Т4.

**Задача Т6**

1. Прийняти від Т5;
2. обчислити
3. надіслати Т5;

**Задача Т7**

1. Ввід C, B, MK
2. Надіслати Т4
3. Прийняти d, MO від Т4
4. Надіслати Т8
5. обчислити
6. прийняти від Т8
7. надіслати Т4

**Задача Т8**

1. Прийняти від Т7;
2. Надіслати Т9;
3. обчислити
4. прийняти від Т9;
5. надіслати Т7.

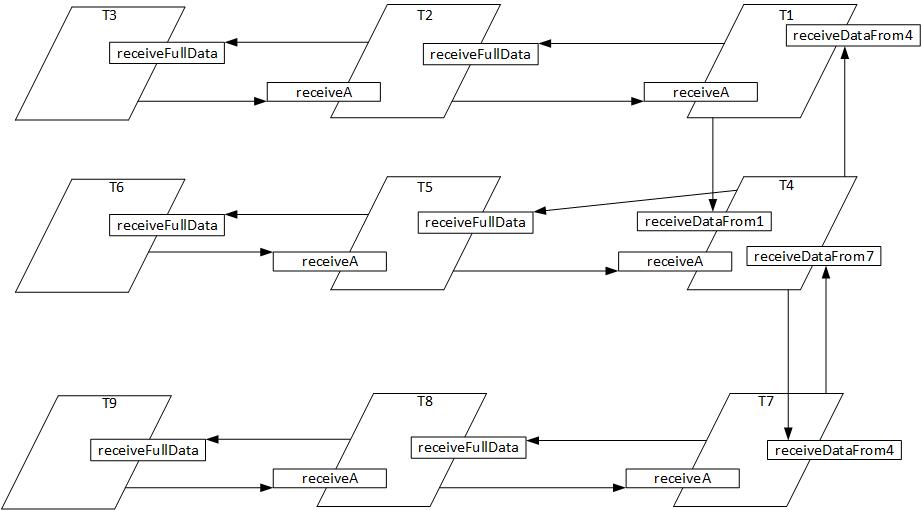
**Задача Т9**

1. Прийняти від Т8;
2. обчислити
3. надіслати Т8;

**Етап 3. Розробка структурної схеми взаємодії задач**

На структурній схемі взаємодії задач введено такі умовні позначення:

1. receiveA – вхід для отримання часткового результату виразу;
2. receiveDataFrom4 – вхід для передачі задачею Т1 даних в Т7 через Т4, або задачею Т7 в Т1 через Т4
3. receiveDataFrom1 – вхід для передачі даних із Т1 в Т4
4. receiveDataFrom7 – вхід для передачі даних із Т7 в Т4
5. receiveFullData – вхід для передачі часткових даних з Т1 та Т7



**Етап 4. Лістинг коду**

GNAT 4.6.4

Copyright 1992-2010, Free Software Foundation, Inc.

Compiling: newMain.adb (source file time stamp: 2016-04-27 01:16:56)

1. with Ada.Text\_IO, Ada.Integer\_text\_IO, Ada.Synchronous\_Task\_Control, MatrixOperations;

2. use Ada.Text\_IO, Ada.Integer\_text\_IO, Ada.Synchronous\_Task\_Control;

3.

4. procedure lab7 is

5.

6. n: Integer; --розмірність матриць та векторів

7. p: Integer := 9; --кількість задач

8. h: Integer;

9.

10. procedure Start is

11.

12. package MatrixOperationsN is new MatrixOperations(N);

13. use MatrixOperationsN;

14.

15. procedure calcMatrixEquation(d,startIndex,endIndex: Integer;

16. Bh:DynamicVector; C:DynamicVector; Res: out DynamicVector; MO:DynamicMatrix; MKh:DynamicMatrix) is

17. MatrixBuffer : DynamicMatrix(MKh'Range);

18. VectorBuffer : DynamicVector(Bh'Range);

19. begin

20. Matrix\_Matrix\_Multiply(MKh,MO,MatrixBuffer);

21. Vector\_Matrix\_Multiply(C,MatrixBuffer,VectorBuffer);

22. Vector\_Vector\_Add(Bh,VectorBuffer,Res,d,startIndex,endIndex);

23. end calcMatrixEquation;

24. Task T1 is

25. entry receiveDataFrom4(Bh,C:DynamicVector; MKh : DynamicMatrix);

26. entry receiveA(Ah:DynamicVector);

27. end T1;

28.

29. Task T7 is

30. entry receiveDataFrom4(d: Integer; MO : DynamicMatrix);

31. entry receiveA(Ah:DynamicVector);

32. end T7;

33.

34. Task T2 is

35. entry receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer);

36. entry receiveA(Ah:DynamicVector);

37. end T2;

38.

39. Task T5 is

40. entry receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer);

41. entry receiveA(Ah:DynamicVector);

42. end T5;

43.

44. Task T8 is

45. entry receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer);

46. entry receiveA(Ah:DynamicVector);

47. end T8;

48.

49. Task T3 is

50. entry receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer);

51. end T3;

52.

53. Task T6 is

54. entry receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer);

55. end T6;

56.

57. Task T9 is

58. entry receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer);

59. end T9;

60.

61. Task T4 is

62. entry receiveDataFrom7(Bh,C:DynamicVector; MKh : DynamicMatrix);

63. entry receiveA(Ah:DynamicVector);

64. entry receiveDataFrom1(d:Integer; MO:DynamicMatrix);

65. end T4;

66.

67. Task body T1 is

68. MO: DynamicMatrix(1..N);

69. MKhi: DynamicMatrix(1..3\*H);

70. d:Integer;

71. Bhi:DynamicVector(1..3\*H);

72. Ci,Ahi:DynamicVector(1..N);

73. begin

74. Put\_Line("Task 1 started");

75. d := Input\_Constant;

76. MO := Input\_Matrix;

77.

78. Put\_Line("T1 sended own data to T4");

79. T4.receiveDataFrom1(d, MO);

80.

81. accept receiveDataFrom4(Bh,C:DynamicVector; MKh : DynamicMatrix) do

82. Put\_Line("T1 received data from T4");

83. Ci := C;

84. Bhi := Bh;

85. MKhi := MKh;

86. end receiveDataFrom4;

87.

88. Put\_Line("T1 sended full data to T2");

89. T2.receiveFullData(Bhi(H+1..3\*H),Ci,MO,MKhi(H+1..3\*H), d);

90.

91. Put\_Line("T1 is calculating matrixEquation");

92. calcMatrixEquation(d,Bhi'First,Bhi'Last,Bhi,Ci,Ahi,MO,MKhi);

93. Put\_Line("T1 is finished calculating matrixEquation");

94.

95.

96. accept receiveA(Ah:DynamicVector) do

97. Put\_Line("T1 received one part of result");

98. Ahi(Ah'Range) := Ah(Ah'Range);

99. end receiveA;

100.

101. accept receiveA(Ah:DynamicVector) do

102. Put\_Line("T1 received one part of result");

103. Ahi(Ah'Range) := Ah(Ah'Range);

104. end receiveA;

105.

106. if N <= 18 then

107. Output\_Vector(Ahi);

108. end if;

109. Put\_Line("Task 1 finished");

110. end T1;

111.

112. Task body T7 is

113. MOi: DynamicMatrix(1..N);

114. MKhi: DynamicMatrix(1..N);

115. di:Integer;

116. Bhi,Ci:DynamicVector(1..N);

117. Ahi:DynamicVector(6\*H+1..N);

118. begin

119. Put\_Line("T7 started");

120. Bhi := Input\_Vector;

121. Ci := Input\_Vector;

122. MKhi := Input\_Matrix;

123.

124. Put\_Line("T7 sends own data to T4");

125. T4.receiveDataFrom7(Bhi(1..6\*H),Ci,MKhi(1..6\*H));

126.

127. Put\_Line("T7 is waiting data from T4");

128. accept receiveDataFrom4(d: Integer; MO : DynamicMatrix) do

129. Put\_Line("T7 received data from T4");

130. di := d;

131. MOi := MO;

132. end receiveDataFrom4;

133.

134. Put\_Line("T7 sends full data to T8");

135. T8.receiveFullData(Bhi(7\*H+1..N),Ci,MOi,MKhi(7\*H+1..N),di);

136.

137. Put\_Line("T7 is calculating matrixEquation");

138. calcMatrixEquation(di,Ahi'First,Ahi'Last,Bhi,Ci,Ahi,MOi,MKhi);

139. Put\_Line("T7 is finished calculating matrixEquation");

140.

141. accept receiveA(Ah:DynamicVector) do

142. Ahi(Ah'Range) := Ah(Ah'Range);

143. Put\_Line("T7 received result data (1)");

144. end receiveA;

145.

146. Put\_Line("Task 7 sends result data to T4");

147. T4.receiveA(Ahi);

148. Put\_Line("Task 7 finished");

149. end T7;

150.

151. Task body T4 is

152. MOi: DynamicMatrix(1..N);

153. MKhi: DynamicMatrix(1..6\*H);

154. di:Integer;

155. Bhi:DynamicVector(1..6\*H);

156. Ahi:DynamicVector(3\*H+1..N);

157. Ci:DynamicVector(1..N);

158. begin

159. Put\_Line("Task 4 started");

160. Put\_Line("T4 is waiting data from T1");

161. accept receiveDataFrom1(d:Integer; MO:DynamicMatrix) do

162. Put\_Line("T4 received data from T1");

163. di := d;

164. MOi := MO;

165. end receiveDataFrom1;

166.

167. Put\_Line("T4 is waiting data from T7");

168. accept receiveDataFrom7(Bh,C:DynamicVector; MKh : DynamicMatrix) do

169. Put\_Line("T4 received data from T7");

170. Bhi:=Bh;

171. Ci:=C;

172. MKhi:=MKh;

173. Put\_Line("T4 finished receiving data from T7");

174. end receiveDataFrom7;

175.

176. Put\_Line("T4 send data of T1 to T7");

177. T7.receiveDataFrom4(di,MOi);

178.

179. Put\_Line("T4 starts sending data of T7 to T1");

180. T1.receiveDataFrom4(Bhi(1..3\*H),Ci, MKhi(1..3\*H));

181. Put\_Line("T4 finished sending data of T7 to T1");

182.

183. Put\_Line("T4 send full data to T5");

184. T5.receiveFullData(Bhi(4\*H+1..6\*H),Ci,MOi, MKhi(4\*H+1..6\*H),di);

185.

186. calcMatrixEquation(di,3\*H+1,4\*H,Bhi,Ci,Ahi,MOi,MKhi);

187.

188. accept receiveA(Ah:DynamicVector) do

189. Ahi(Ah'Range) := Ah(Ah'Range);

190. Put\_Line("T4 received result data (1)");

191. end receiveA;

192.

193. accept receiveA(Ah:DynamicVector) do

194. Ahi(Ah'Range) := Ah(Ah'Range);

195. Put\_Line("T4 received result data (2)");

196. end receiveA;

197.

198. Put\_Line("T4 sends result data to T1");

199. T1.receiveA(Ahi);

200. Put\_Line("Task 4 finished");

201. end T4;

202.

203. task body T2 is

204. MOi: DynamicMatrix(1..N);

205. MKhi: DynamicMatrix(H+1..3\*H);

206. di:Integer;

207. Bhi:DynamicVector(H+1..3\*H);

208. Ahi:DynamicVector(H+1..3\*H);

209. Ci:DynamicVector(1..N);

210. begin

211. Put\_Line("Task 2 started");

212. accept receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer) do

213. Bhi := Bh;

214. Ci := C;

215. MOi := MO;

216. MKhi := Mkh;

217. di := d;

218. Put\_Line("Task 2 received full data from T1");

219. end receiveFullData;

220.

221. Put\_Line("Task 2 sends full data to T3");

222. T3.receiveFullData(Bhi(2\*H+1..3\*H),Ci,MOi,MKhi(2\*H+1..3\*H),di);

223.

224. calcMatrixEquation(di,Bhi'First,Bhi'Last,Bhi,Ci,Ahi,MOi,MKhi);

225.

226. Put\_Line("Task 2 finished calculating matrixEquation");

227. accept receiveA(Ah:DynamicVector) do

228. Ahi(Ah'Range) := Ah(Ah'Range);

229. end receiveA;

230.

231. Put\_Line("T2 sends result data to T1");

232. T1.receiveA(Ahi);

233. Put\_Line("Task 2 finished");

234. end T2;

235.

236. task body T3 is

237. MOi: DynamicMatrix(1..N);

238. MKhi: DynamicMatrix(2\*H+1..3\*H);

239. di:Integer;

240. Bhi,Ahi:DynamicVector(2\*H+1..3\*H);

241. Ci:DynamicVector(1..N);

242. begin

243. Put\_Line("Task 3 started");

244. accept receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer) do

245. Bhi := Bh;

246. Ci := C;

247. MOi := MO;

248. MKhi := Mkh;

249. di := d;

250. end receiveFullData;

251.

252. calcMatrixEquation(di,Bhi'First,Bhi'Last,Bhi,Ci,Ahi,MOi,MKhi);

253.

254. Put\_Line("T3 sends result data to T2");

255. T2.receiveA(Ahi);

256. Put\_Line("Task 3 finished");

257. end T3;

258.

259. task body T5 is

260. MOi: DynamicMatrix(1..N);

261. MKhi: DynamicMatrix(4\*H+1..6\*H);

262. di:Integer;

263. Bhi,Ahi:DynamicVector(4\*H+1..6\*H);

264. Ci:DynamicVector(1..N);

265. begin

266. Put\_Line("Task 5 started");

267. accept receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer) do

268. Bhi := Bh;

269. Ci := C;

270. MOi := MO;

271. MKhi := Mkh;

272. di := d;

273. end receiveFullData;

274.

275. T6.receiveFullData(Bhi(5\*H+1..6\*H),Ci,MOi,MKhi(5\*H+1..6\*H),di);

276.

277. calcMatrixEquation(di,Bhi'First,Bhi'Last,Bhi,Ci,Ahi,MOi,MKhi);

278.

279. accept receiveA(Ah:DynamicVector) do

280. Ahi(Ah'Range) := Ah(Ah'Range);

281. end receiveA;

282.

283. Put\_Line("T5 sends result data to T4");

284. T4.receiveA(Ahi);

285. Put\_Line("Task 5 finished");

286. end T5;

287.

288. task body T6 is

289. MOi: DynamicMatrix(1..N);

290. MKhi: DynamicMatrix(5\*H+1..6\*H);

291. di:Integer;

292. Bhi:DynamicVector(5\*H+1..6\*H);

293. Ci:DynamicVector(1..N);

294. Ahi:DynamicVector(5\*H+1..6\*h);

295. begin

296. Put\_Line("Task 6 started");

297. accept receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer) do

298. Bhi := Bh;

299. Ci := C;

300. MOi := MO;

301. MKhi := Mkh;

302. di := d;

303. end receiveFullData;

304.

305. calcMatrixEquation(di,Bhi'First,Bhi'Last,Bhi,Ci,Ahi,MOi,MKhi);

306.

307. Put\_Line("T6 sends result data to T5");

308. T5.receiveA(Ahi);

309. Put\_Line("Task 6 finished");

310. end T6;

311.

312. task body T8 is

313. MOi: DynamicMatrix(1..N);

314. MKhi: DynamicMatrix(7\*H+1..N);

315. di:Integer;

316. Bhi:DynamicVector(7\*H+1..N);

317. Ci:DynamicVector(1..N);

318. Ahi:DynamicVector(7\*H+1..N);

319. begin

320. Put\_Line("Task 8 started");

321. accept receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer) do

322. Bhi := Bh;

323. Ci := C;

324. MOi := MO;

325. MKhi := Mkh;

326. di := d;

327. end receiveFullData;

328.

329. T9.receiveFullData(Bhi(8\*H+1..N),Ci,MOi,MKhi(8\*H+1..N),di);

330.

331. calcMatrixEquation(di,Bhi'First,Bhi'Last,Bhi,Ci,Ahi,MOi,MKhi);

332.

333. accept receiveA(Ah:DynamicVector) do

334. Ahi(Ah'Range) := Ah(Ah'Range);

335. end receiveA;

336. Put\_Line("T8 received result data from T9");

337.

338. Put\_Line("T8 sends result data to T7");

339. T7.receiveA(Ahi);

340. Put\_Line("Task 8 finished");

341. end T8;

342.

343. task body T9 is

344. MOi: DynamicMatrix(1..N);

345. MKhi: DynamicMatrix(8\*H+1..N);

346. di:Integer;

347. Bhi:DynamicVector(8\*H+1..N);

348. Ci:DynamicVector(1..N);

349. Ahi:DynamicVector(8\*H+1..N);

350. begin

351. Put\_Line("Task 9 started");

352. accept receiveFullData(Bh,C:DynamicVector;MO,MKh:DynamicMatrix; d:Integer) do

353. Bhi := Bh;

354. Ci := C;

355. MOi := MO;

356. MKhi := Mkh;

357. di := d;

358. end receiveFullData;

359.

360. calcMatrixEquation(di,Bhi'First,Bhi'Last,Bhi,Ci,Ahi,MOi,MKhi);

361.

362. Put\_Line("T9 sends result data to T8");

363. T8.receiveA(Ahi);

364. Put\_Line("Task 9 finished");

365. end T9;

366.

367.

368. begin

369. null;

370. end Start;

371. begin

372. Put("Program started");

373. New\_line;

374. Put("Input N");

375. New\_line;

376. Get(n);

377. h:= n/p;

378. Start;

379. end;

379 lines: 0 error, 0 warning

GNAT 4.6.4

Copyright 1992-2010, Free Software Foundation, Inc.

Compiling: MatrixOperations.adb (source file time stamp: 2016-04-27 00:18:52)

1. ------------------------------------------------------------------

2. -- File: MatrixOperations.adb --

3. -- Author: Kakherskyi Oleh, group IP-31 --

4. -- Date: 12.09.2015 --

5. ------------------------------------------------------------------

6. with Ada.Text\_IO, Ada.Integer\_Text\_IO;

7. use Ada.Text\_IO, Ada.Integer\_Text\_IO;

8.

9. package body MatrixOperations is

10.

11. function Input\_Vector return DynamicVector is

12. V: DynamicVector(1..N);

13. begin

14. for I in V'Range loop

15. V(I) := 1;

16. end loop;

17. return V;

18. end Input\_Vector;

19.

20. function Input\_Constant return Integer is

21. begin

22. return 1;

23. end Input\_Constant;

24.

25. function Input\_Matrix return DynamicMatrix is

26. Result\_Matrix: DynamicMatrix(1 .. N);

27. begin

28. for I in Result\_Matrix'Range loop

29. for J in 1..N loop

30. Result\_Matrix(I)(J) := 1;

31. end loop;

32. end loop;

33. return Result\_Matrix;

34. end Input\_Matrix;

35.

36. -------------------

37. -- Output\_Vector --

38. -------------------

39.

40. procedure Output\_Vector (V : in DynamicVector) is

41. begin

42. New\_Line;

43. for I in V'Range loop

44. Put(Item => V(I), Width => 5);

45. end loop;

46. New\_Line;

47. end Output\_Vector;

48.

49.

50. procedure Matrix\_Matrix\_Multiply

51. (Left : in DynamicMatrix;

52. Right : in DynamicMatrix;

53. Result\_Matrix: out DynamicMatrix) is

54. begin

55. for i in Left'Range loop

56. for J in 1..N loop

57. Result\_Matrix(I)(J) := 0;

58. for K in Right'Range loop

59. Result\_Matrix(I)(J) := Result\_Matrix(I)(J) + Left(I)(K) \* Right(K)(J);

60. end loop;

61. end loop;

62. end loop;

63. end Matrix\_Matrix\_Multiply;

64.

65.

66. procedure Vector\_Matrix\_Multiply

67. (Left : in DynamicVector;

68. Right : in DynamicMatrix;

69. Result\_Vector: out DynamicVector) is

70. begin

71. for J in Result\_Vector'Range loop

72. Result\_Vector(j) := 0;

73. begin

74. for K in 1..N loop

75. Result\_Vector(J) := Result\_Vector(J) + Left(K) \* Right(J)(K);

76. end loop;

77. end;

78. end loop;

79. end Vector\_Matrix\_Multiply;

80.

81. procedure addVectors(Left: DynamicVector; Right: DynamicVector; Result\_Vector: out DynamicVector; d : Integer) is

82. begin

83. for J in Right'Range loop

84. Result\_Vector(J) := d \* Left(J) + Right(J);

85. end loop;

86. end addVectors;

87.

88. procedure Vector\_Vector\_Add(Left: DynamicVector; Right: DynamicVector; Result\_Vector: out DynamicVector; d,startIndex,EndIndex : Integer) is

89. begin

90. for J in startIndex..EndIndex loop

91. Result\_Vector(J) := d \* Left(J) + Right(J);

92. end loop;

93. end Vector\_Vector\_Add;

94.

95. end MatrixOperations;

95 lines: 0 error, 0 warning