1. Структурна схема ПКС ЛП
2. Схема алгоритму головної програми зі вказанням паралельних ділянок для ПРГ2
3. Схема алгоритму процесів для програми ПРГ2
4. Лістинг програми ПРГ2

1. with Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Synchronous\_Task\_Control, Ada.Command\_Line;

2. use Ada.Text\_IO, Ada.Integer\_Text\_IO, Ada.Synchronous\_Task\_Control, Ada.Command\_Line;

3.

4. procedure Main is

5. --N : Integer := Integer'Value(Argument(1));

6. N : Integer := 4;

7. P: integer := 4; --Number of tasks

8. H: integer := N / P; --Size of subvector

9.

10. type General\_Vector is array(integer range <>) of integer;

11. subtype Vector is General\_Vector(1..N);

12.

13. type General\_Matrix is array(integer range <>) of Vector;

14. subtype Matrix is General\_Matrix(1..N);

15.

16. procedure output(A : Vector) is

17. begin

18. New\_Line;

19. for I in 1..N loop

20. Put(Item => A(I), Width => 6);

21. put("|");

22. end loop;

23. New\_Line;

24. end output;

25.

26. procedure output (MA : in Matrix) is

27. begin

28. New\_Line;

29. for I in 1..N loop

30. for J in 1..N loop

31. Put(Item => MA(i)(j), Width => 6);

32. put("||");

33. end loop;

34. New\_line;

35. end loop;

36. New\_Line;

37. end Output;

38.

39. --Task specification T1------------------------------------

40. task T1 is

41. entry Send\_Matrixes(MX: in Matrix; MZh: in General\_Matrix);

42. entry Send\_D2(Dh: in General\_Vector);

43. entry Send\_D4(D2h: in General\_Vector);

44. entry Send\_A2(Ah: in General\_Vector);

45. entry Send\_A4(A2h: in General\_Vector);

46. end T1;

47. -----------------------------------------------------------

48.

49. --Task specification T2------------------------------------

50. task T2 is

51. entry Send\_Vectors(Bh: in General\_Vector; C: in Vector);

52. entry Send\_Matrix(MOh: in General\_Matrix);

53. entry Send\_D(D: in Vector);

54. end T2;

55. -----------------------------------------------------------

56.

57. --Task specification T3------------------------------------

58. task T3 is

59. entry Send\_Vectors(Bh: in General\_Vector; C: in Vector);

60. entry Send\_Matrix(MOh: in General\_Matrix);

61. entry Send\_Matrixes(MX: in Matrix; MZ2h: in General\_Matrix);

62. entry Send\_D(D: in Vector);

63. end T3;

64. -----------------------------------------------------------

65.

66. --Task specification T4------------------------------------

67. task T4 is

68. entry Send\_Vectors(B2h: in General\_Vector; C: in Vector);

69. entry Send\_Matrix(MO2h: in General\_Matrix);

70. entry Send\_Matrixes(MX: in Matrix; MZh: in General\_Matrix);

71. entry Send\_Dh(Dh: in General\_Vector);

72. entry Send\_D(D: in Vector);

73. entry Send\_Ah(Ah: in General\_Vector);

74. end T4;

75. -----------------------------------------------------------

76.

77. --Task body T1---------------------------------------------

78. task body T1 is

79. A, B, C, D, X1 : Vector;

80. MM1, MO, MX1, MZ1: Matrix;

81. begin

82. put\_line("T1 started");

83.

84. -- 1. Input B, C, MO.

85. for i in 1..N loop

86. B(i) := 1;

87. C(i) := 1;

88.

89. A(i) := 0;

90. D(i) := 0;

91. X1(i) := 0;

92. for j in 1..N loop

93. MO(i)(j) := 1;

94.

95. MM1(i)(j) := 0;

96. MX1(i)(j) := 0;

97. MZ1(i)(j) := 0;

98. end loop;

99. end loop;

100.

101. --put\_line("T1\_AFTER\_1");

102.

103. -- 2. Receiving MX, MZh.

104. accept Send\_Matrixes(MX: in Matrix; MZh: in General\_Matrix) do

105. MX1 := MX;

106. MZ1(MZh'Range) := MZh;

107. end Send\_Matrixes;

108.

109. --put\_line("T1\_AFTER\_2");

110.

111. -- 3.Sending Bh,C,MOh to other tasks.

112. T2.Send\_Vectors(B(H+1..2\*H), C);

113. T2.Send\_Matrix(MO(H+1..2\*H));

114. T4.Send\_Vectors(B(2\*H+1..N), C);

115. T4.Send\_Matrix(MO(2\*H+1..N));

116.

117. --put\_line("T1\_AFTER\_3");

118.

119. -- 4. Calc X1:=C\*MO1

120. for I in 1..H loop

121. X1(I) := 0;

122. for J in 1..N loop

123. X1(I) := X1(I) + C(J) \* MO(I)(J);

124. end loop;

125. end loop;

126.

127. --put\_line("T1\_AFTER\_4");

128.

129. -- 5. Calc MM1:=MX\*MZ1

130. for i in 1..N loop

131. for J in 1..H loop

132. MM1(I)(J) := 0;

133. for K in 1..N loop

134. MM1(I)(J) := MM1(I)(J) + MX1(I)(K) \* MZ1(K)(J);

135. end loop;

136. end loop;

137. end loop;

138.

139. --put\_line("T1\_AFTER\_5");

140.

141. -- 6. D1:=B1+X1

142. for I in 1..H loop

143. D(I) := B(I) + X1(I);

144. end loop;

145.

146. --put\_line("T1\_AFTER\_6");

147.

148. -- 7. Receive D3h

149. accept Send\_D2 (Dh : in General\_Vector) do

150. D(Dh'Range) := Dh;

151. end Send\_D2;

152.

153. accept Send\_D4 (D2h : in General\_Vector) do

154. D(D2h'Range) := D2h;

155. end Send\_D4;

156.

157. --put\_line("T1\_AFTER\_7");

158.

159. -- 8. Sending D to T2, T4

160. T2.Send\_D(D);

161. T4.Send\_D(D);

162.

163. --put\_line("T1\_AFTER\_8");

164.

165. -- 9. Calculate A1:=D\*MM1

166. for I in 1..H loop

167. begin

168. A(I) := 0;

169. for J in 1..N loop

170. A(I) := A(I) + D(J) \* MM1(J)(I);

171. end loop;

172. end;

173. end loop;

174.

175. --put\_line("T1\_AFTER\_9");

176.

177. -- 10. Receive 3Ah

178. accept Send\_A2 (Ah : in General\_Vector) do

179. A(Ah'Range) := Ah;

180. end Send\_A2;

181. accept Send\_A4 (A2h : in General\_Vector) do

182. A(A2h'Range) := A2h;

183. end Send\_A4;

184.

185. --put\_line("T1\_AFTER\_10");

186.

187. -- 11. Output A.

188.

189. if(N < 10) then

190. output(A);

191. end if;

192.

193. put\_line("T1 finished");

194. end T1;

195. -----------------------------------------------------------

196.

197. --Task body T2---------------------------------------------

198. task body T2 is

199. A2, B2, C2, D2, X2 : Vector;

200. MM2, MO2, MX2, MZ2: Matrix;

201.

202. begin

203. put\_line("T2 started");

204.

205. -- 1. Input MX, MZ.

206. for I in 1..N loop

207. A2(I) := 0;

208. B2(I) := 0;

209. C2(I) := 0;

210. D2(I) := 0;

211. X2(I) := 0;

212. for J in 1..N loop

213. MX2(I)(J) := 1;

214. MZ2(I)(J) := 1;

215.

216. MM2(I)(J) := 0;

217. MO2(I)(J) := 0;

218. end loop;

219. end loop;

220.

221. --put\_line("T2\_AFTER\_1");

222.

223. -- 2. Sending MX, MZ.

224. T1.Send\_Matrixes(MX2, MZ2(1..H));

225.

226. --put\_line("T2\_AFTER\_2");

227.

228. -- 3. Receiving B, C, MO.

229. accept Send\_Vectors (Bh : in General\_Vector; C : in Vector) do

230. B2(Bh'Range) := Bh;

231. C2 := C;

232. end Send\_Vectors;

233. accept Send\_Matrix(MOh: in General\_Matrix) do

234. MO2(MOh'Range) := MOh;

235. end Send\_Matrix;

236.

237. --put\_line("T2\_AFTER\_3");

238.

239. -- 4. Sending MX, MZ.

240. T3.Send\_Matrixes(MX2, MZ2(2\*H+1..N));

241.

242. --put\_line("T2\_AFTER\_4");

243.

244. -- 5. Calc X2:=C\*MO2

245. for I in H+1..2\*H loop

246. X2(I) := 0;

247. for J in 1..N loop

248. X2(I) := X2(I) + C2(J) \* MO2(I)(J);

249. end loop;

250. end loop;

251.

252. --put\_line("T2\_AFTER\_5");

253.

254. -- 6. Calc MM2:=MX\*MZ2

255. for i in 1..N loop

256. for J in H+1..2\*H loop

257. MM2(I)(J) := 0;

258. for K in 1..N loop

259. MM2(I)(J) := MM2(I)(J) + MX2(I)(K) \* MZ2(K)(J);

260. end loop;

261. end loop;

262. end loop;

263.

264. --put\_line("T2\_AFTER\_6");

265.

266. -- 7. D2:=B2+X2

267. for I in H+1..2\*H loop

268. D2(I) := B2(I) + X2(I);

269. end loop;

270.

271. --put\_line("T2\_AFTER\_7");

272.

273. -- 8. Sending Dh.

274. T1.Send\_D2(D2(H+1..2\*H));

275.

276. --put\_line("T2\_AFTER\_8");

277.

278. -- 9. Receiving D.

279. accept Send\_D (D : in Vector) do

280. D2 := D;

281. end Send\_D;

282.

283. --put\_line("T2\_AFTER\_9");

284.

285. -- 10. A2:=D\*MM2

286. for I in H+1..2\*H loop

287. begin

288. A2(I) := 0;

289. for J in 1..N loop

290. A2(I) := A2(I) + D2(J) \* MM2(I)(J);

291. end loop;

292. end;

293. end loop;

294.

295. --put\_line("T2\_AFTER\_10");

296.

297. -- 11. Sending Ah

298. T1.Send\_A2(A2(H+1..2\*H));

299.

300. put\_line("T2 finished");

301. end T2;

302. -----------------------------------------------------------

303.

304. --Task body T3---------------------------------------------

305. task body T3 is

306. A3, B3, C3, D3, X3 : Vector;

307. MM3, MO3, MX3, MZ3: Matrix;

308. begin

309. put\_line("T3 started");

310.

311. -- 1. Init.

312. for I in 1..N loop

313. for J in 1..N loop

314. MX3(I)(J) := 0;

315. MZ3(I)(J) := 0;

316.

317. A3(I) := 0;

318. B3(I) := 0;

319. C3(I) := 0;

320. D3(I) := 0;

321. X3(I) := 0;

322. MM3(I)(J) := 0;

323. MO3(I)(J) := 0;

324. end loop;

325. end loop;

326.

327. --put\_line("T3\_AFTER\_1");

328.

329. accept Send\_Matrixes (MX : in Matrix; MZ2h : in General\_Matrix) do

330. MX3 := MX;

331. MZ3(MZ2h'Range) := MZ2h;

332. end Send\_Matrixes;

333.

334. T4.Send\_Matrixes(MX3,MZ3(3\*H+1..N));

335.

336. -- 2. Receiving All

337. accept Send\_Vectors (Bh : in General\_Vector; C : in Vector) do

338. B3(Bh'Range) := Bh;

339. C3 := C;

340. end Send\_Vectors;

341.

342. accept Send\_Matrix (MOh : in General\_Matrix) do

343. MO3(MOh'Range) := MOh;

344. end Send\_Matrix;

345.

346. --put\_line("T3\_AFTER\_2");

347.

348. -- 3. Calc X3:=C\*MO3

349. for I in 2\*H+1..3\*H loop

350. X3(I) := 0;

351. for J in 1..N loop

352. X3(I) := X3(I) + C3(J) \* MO3(I)(J);

353. end loop;

354. end loop;

355.

356. --put\_line("T3\_AFTER\_3");

357. --output(MX3);

358.

359. -- 4. Calc MM3:=MX\*MZ3

360. for I in 1..2\*H loop

361. for J in 2\*H+1..3\*H loop

362. MM3(I)(J) := 0;

363. for K in 1..N loop

364. MM3(I)(J) := MM3(I)(J) + MX3(I)(K) \* MZ3(K)(J);

365. end loop;

366. end loop;

367. end loop;

368.

369. --put\_line("T3\_AFTER\_4");

370.

371. -- 5. D3:=B3+X3

372. for I in 2\*H+1..3\*H loop

373. D3(I) := B3(I) + X3(I);

374. end loop;

375.

376. --put\_line("T3\_AFTER\_5");

377.

378. -- 6. Sending Dh.

379. T4.Send\_Dh(D3(2\*H+1..3\*H));

380.

381. --put\_line("T3\_AFTER\_6");

382.

383. -- 7. Receiving D.

384. accept Send\_D (D : in Vector) do

385. D3 := D;

386. end Send\_D;

387.

388. --put\_line("T3\_AFTER\_7");

389.

390. -- 8. A2:=D\*MM2

391. for I in 2\*H+1..3\*H loop

392. begin

393. A3(I) := 0;

394. for J in 1..N loop

395. A3(I) := A3(I) + D3(J) \* MM3(J)(I);

396. end loop;

397. end;

398. end loop;

399.

400. --put\_line("T3\_AFTER\_8");

401.

402. -- 9. Sending Ah

403. T4.Send\_Ah(A3(2\*H+1..3\*H));

404.

405. put\_line("T3 finished");

406. end T3;

407. -----------------------------------------------------------

408.

409. --Task body T4---------------------------------------------

410. task body T4 is

411. A4, B4, C4, D4, X4 : Vector;

412. MM4, MO4, MX4, MZ4: Matrix;

413. begin

414. put\_line("T4 started");

415.

416. -- 1. Init.

417. for I in 1..N loop

418. for J in 1..N loop

419. MX4(I)(J) := 0;

420. MZ4(I)(J) := 0;

421.

422. A4(I) := 0;

423. B4(I) := 0;

424. C4(I) := 0;

425. D4(I) := 0;

426. X4(I) := 0;

427. MM4(I)(J) := 0;

428. MO4(I)(J) := 0;

429. end loop;

430. end loop;

431.

432. --put\_line("T4\_AFTER\_1");

433.

434. -- 2. Receiving All

435. accept Send\_Vectors (B2h : in General\_Vector; C : in Vector) do

436. B4(B2h'Range) := B2h;

437. C4 := C;

438. end Send\_Vectors;

439.

440. accept Send\_Matrix (MO2h : in General\_Matrix) do

441. MO4(MO2h'Range) := MO2h;

442. end Send\_Matrix;

443.

444. accept Send\_Matrixes (MX : in Matrix; MZh : in General\_Matrix) do

445. MX4 := MX;

446. MZ4(MZh'Range) := MZh;

447. end Send\_Matrixes;

448.

449. T3.Send\_Vectors(B4(2\*H+1..3\*H), C4);

450. T3.Send\_Matrix(MO4(2\*H+1..3\*H));

451.

452. --put\_line("T4\_AFTER\_2");

453.

454. -- 3. Calc X4:=C\*MO4

455. for I in 3\*H+1..N loop

456. X4(I) := 0;

457. for J in 1..N loop

458. X4(I) := X4(I) + C4(J) \* MO4(I)(J);

459. end loop;

460. end loop;

461.

462. --put\_line("T4\_AFTER\_3");

463.

464. -- 4. Calc MM4:=MX\*MZ4

465. for I in 1..N loop

466. for J in 3\*H+1..N loop

467. MM4(I)(J) := 0;

468. for K in 1..N loop

469. MM4(I)(J) := MM4(I)(J) + MX4(I)(K) \* MZ4(K)(J);

470. end loop;

471. end loop;

472. end loop;

473.

474. --put\_line("T4\_AFTER\_4");

475.

476. -- 5. D4:=B4+X4

477. for I in 3\*H+1..N loop

478. D4(I) := B4(I) + X4(I);

479. end loop;

480.

481. accept Send\_Dh (Dh : in General\_Vector) do

482. D4(Dh'Range) := Dh;

483. end Send\_Dh;

484.

485. --put\_line("T4\_AFTER\_5");

486.

487. -- 6. Sending Dh.

488. T1.Send\_D4(D4(2\*H+1..N));

489.

490. --put\_line("T4\_AFTER\_6");

491.

492. -- 7. Receiving D.

493. accept Send\_D (D : in Vector) do

494. D4 := D;

495. end Send\_D;

496.

497. T3.Send\_D(D4);

498.

499. --put\_line("T4\_AFTER\_7");

500.

501. -- 8. A4:=D\*MM4

502. for I in 3\*H+1..N loop

503. begin

504. A4(I) := 0;

505. for J in 1..N loop

506. A4(I) := A4(I) + D4(J) \* MM4(J)(I);

507. end loop;

508. end;

509. end loop;

510.

511. accept Send\_Ah (Ah : in General\_Vector) do

512. A4(Ah'Range) := Ah;

513. end Send\_Ah;

514.

515. --put\_line("T4\_AFTER\_8");

516.

517. -- 9. Sending Ah

518. T1.Send\_A4(A4(2\*H+1..N));

519.

520. put\_line("T4 finished");

521. end T4;

522. -----------------------------------------------------------

523.

524. begin

525. null;

526. end Main;

1. Код скрипта для тестування ПРГ2

1. import time, subprocess, sys

2.

3. COMMAND = ' '.join(sys.argv[1:])

4.

5. def measure\_time(affinity, command):

6. start\_t = time.time()

7.

8. subp = subprocess.Popen('start /AFFINITY 0x%X %s' % (pow(2, affinity) - 1, command), shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

9. stderr, stdout = subp.communicate()

10. return\_code = subp.returncode

11. subp.wait()

12.

13. end\_t = time.time()

14.

15. return (end\_t - start\_t)

16.

17. for i in xrange(1, 5):

18. print 'Running %s with %d processors\n' % (COMMAND, i)

19. time\_elapsed = measure\_time(i, COMMAND)

20. print 'Time elapsed: %f' % time\_elapsed