Національний технічний університет України

«Київський Політехнічний Інститут»

Факультет інформатики і обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №7**

**З предмету «Паралельні та розподілені обчислення»**

**Ада. Рандеву**

Виконав:

Студент  
IІІ курсу ФІОТ  
групи ІО-12  
Бута С. О.

Залікова книжка №1205

Київ-2014

**Техническое задание**

1. Структура ПКС с ЛП: 
2. Задача: .
3. Язык программирования: Ада.
4. Средства взаимодействия задач: рандеву.

**Выполнение работы**

1. Разработка параллельного математического алгоритма.
2. ai = max(ZH);
3. a = max(a, ai);
4. AH = a\*BH – E\*(MO\*MKH); OP: a, E, MO;
5. Разработка алгоритмов процессов.

**Задача Т1:**

1. Ввод Z, E, MO
2. Передать ZN-H, E, MO в T2
3. Принять BH, MKH из Т2
4. Счёт1
   1. a1 = max(ZH)
5. Передать a1 в T2
6. Принять a из T2
7. Счёт2
   1. AH = a\*BH – E\*(MO\*MKH)
8. Передать AH в T2

**Задача Т2:**

1. Принять ZN-H, E, MO из T1
2. Передать ZN-2H, E, MO в T3
3. Принять B2H, MK2H из Т3
4. Передать BH, MKH в Т1
5. Счёт1
   1. а2 = max(ZH)
6. Принять a1 из T1
7. Счёт2
   1. a = max(a2, a1)
8. Передать a в T3
9. Принять a из T3
10. Передать a в T1
11. Счёт3
    1. AH = a\*BH – E\*(MO\*MKH)
12. Принять AH из T1
13. Передать A2Н в T3

**Задача Т3:**

1. Принять ZN-2H, E, MO из T2
2. Передать ZN-4H, E, MO в T4
3. Принять BN-3H, MKN-3H из Т4
4. Передать B2H, MK2H в Т2
5. Передать BH, MKH, ZH, E, MO в Т6
6. Счёт1
   1. а3 = max(ZH)
7. Принять a2 из T2
8. Принять a4 из T4
9. Принять a6 из T6
10. Счёт2
    1. a = max(a2, a3, a6, a4)
11. Передать a в T2
12. Передать a в T4
13. Передать a в T6
14. Счёт3
    1. AH = a\*BH – E\*(MO\*MKH)
15. Принять A2Н из T2
16. Принять AH из T6
17. Передать A4Н в T4

**Задача Т4:**

1. Принять BN-H, MKN-H из Т5
2. Передать BN-3H, MKN-3H в Т3
3. Принять ZN-4H, E, MO из T3
4. Передать ZH, E, MO в T5
5. Передать BH, MKH, ZH, E, MO в Т7
6. Счёт1
   1. А4 = max(ZH)
7. Принять a5 из T5
8. Принять a7 из T7
9. Счёт2
   1. a = max(a4, a5, a7)
10. Передать a в T3
11. Принять a из T3
12. Передать a в T5
13. Передать a в T7
14. Счёт3
    1. AH = a\*BH – E\*(MO\*MKH)
15. Принять A4Н из T3
16. Принять AH из T7
17. Передать A6Н в T5

**Задача Т5:**

1. Ввод B, MK
2. Передать BN-H, MKN-H в Т4
3. Принять ZH, E, MO
4. Счёт1
   1. а5 = max(ZH)
5. Передать a5 в T4
6. Принять a из T4
7. Счёт2
   1. AH = a\*BH – E\*(MO\*MKH)
8. Принять AN-H из T4
9. Вывод А

**Задача Т6:**

1. Принять BH, MKH, ZH, E, MO из Т3
2. Счёт1
   1. а6 = max(ZH)
3. Передать a6 в T3
4. Принять a из T3
5. Счёт3
   1. AH = a\*BH – E\*(MO\*MKH)
6. Передать AН в T3

**Задача Т7:**

1. Принять BH, MKH, ZH, E, MO из Т4
2. Счёт1
   1. а7 = max(ZH)
3. Передать a7 в T4
4. Принять a из T4
5. Счёт3
   1. AH = a\*BH – E\*(MO\*MKH)
6. Передать AН в T4

**Разработка схемы взаимодействия процессов:**

**Разработка программы.**

**Листинг**:

GNAT GPL 2014 (20140331)

Copyright 1992-2014, Free Software Foundation, Inc.

Compiling: C:\Users\Serhiy\Documents\adaprjs\Lab7\main.adb (source file time stamp: 2014-06-05 09:46:22)

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

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

3.

4. -------------------------------------------

5. --Лабораторная работа №7. Ада. Рандеву

6. -- A = max(Z)\*B - C\*(MO\*MK)

7. -- Бута С.О.

8. -- 04.06.2014

9. -------------------------------------------

10. procedure Main is

11. N: Integer := 7;

12. P: Integer := 7;

13. H: Integer := N / P;

14.

15. type Vector is array(Integer range<>) of Integer;

16. subtype Vector\_H is Vector(1..H);

17. subtype Vector\_2H is Vector(1..2\*H);

18. subtype Vector\_3H is Vector(1..3\*H);

19. subtype Vector\_4H is Vector(1..4\*H);

20. subtype Vector\_5H is Vector(1..5\*H);

21. subtype Vector\_6H is Vector(1..6\*H);

22. subtype Vector\_N is Vector(1..N);

23.

24. type Matrix is array(Integer range<>) of Vector\_N;

25. subtype Matrix\_H is Matrix(1..H);

26. subtype Matrix\_2H is Matrix(1..2\*H);

27. subtype Matrix\_3H is Matrix(1..3\*H);

28. subtype Matrix\_4H is Matrix(1..4\*H);

29. subtype Matrix\_5H is Matrix(1..5\*H);

30. subtype Matrix\_6H is Matrix(1..6\*H);

31. subtype Matrix\_N is Matrix(1..N);

32.

33. procedure VectorInput(V: out Vector\_N) is

34. begin

35. for i in 1..N loop

36. V(i) := 1;

37. end loop;

38. end VectorInput;

39.

40.

41. procedure VectorOutput(V: in Vector\_N) is

42. begin

43. for i in 1..(N-1) loop

44. put(V(i));

45. put(" ");

46. end loop;

47. put(V(N));

48. Put\_Line("");

49. end VectorOutput;

50.

51.

52. procedure MatrixInput(M: out Matrix\_N) is

53. begin

54. for col in 1..N loop

55. for row in 1..N loop

56. M(col)(row) := 1;

57. end loop;

58. end loop;

59. end MatrixInput;

60.

61.

62. function Max(x, y: in Integer) return Integer is

63. begin

64. if ( x > y ) then

65. return x;

66. end if;

67. return y;

68. end Max;

69.

70.

71. function Calc1(Z: in Vector\_H) return Integer is

72. m: Integer;

73. begin

74. m := Integer'First;

75.

76. for i in 1..H loop

77. if m < Z(H) then

78. m := Z(H);

79. end if;

80. end loop;

81.

82. return m;

83. end Calc1;

84.

85.

86. procedure Calc3(m: Integer; B: in Vector\_H; E: in Vector\_N;

87. MO: in Matrix\_N; MK: in Matrix\_H; A: out Vector\_H) is

88. sum1, sum2: Integer;

89. begin

90. for k in 1..H loop

91. sum1 := 0;

92.

93. for p in 1..N loop

94. sum2 := 0;

95.

96. for q in 1..N loop

97. sum2 := sum2 + MO(q)(p) \* MK(k)(q);

98. end loop;

99.

100. sum1 := sum1 + E(p) \* sum2;

101. end loop;

102.

103. A(k) := B(k)\* m - sum1;

104. end loop;

105. end Calc3;

106.

107.

108. task Task1 is

109. entry SetData(B: in Vector\_H; MK: in Matrix\_H);

110. entry GetData(Z: out Vector\_6H; E: out Vector\_N; MO: out Matrix\_N);

111.

112. entry Get\_ai(mi: out Integer);

113. entry Set\_a(m: in Integer);

114.

115. entry Res\_A(A: out Vector\_H);

116. end Task1;

117.

118. task Task2 is

119. entry SetData(B: in Vector\_2H; MK: in Matrix\_2H);

120. entry GetData(Z: out Vector\_5H; E: out Vector\_N; MO: out Matrix\_N);

121.

122. entry Get\_ai(mi: out Integer);

123. entry Set\_a(m: in Integer);

124.

125. entry Res\_A(A: out Vector\_2H);

126. end Task2;

127.

128. task Task3 is

129. entry SetData(B: in Vector\_4H; MK: in Matrix\_4H);

130. entry GetData(Z: out Vector\_3H; E: out Vector\_N; MO: out Matrix\_N);

131.

132. entry Res\_A(A: out Vector\_4H);

133. end Task3;

134.

135. task Task4 is

136. entry SetData(B: in Vector\_6H; MK: in Matrix\_6H);

137. entry GetData(Z: out Vector\_H; E: out Vector\_N; MO: out Matrix\_N);

138.

139. entry Get\_ai(mi: out Integer);

140. entry Set\_a(m: in Integer);

141.

142. entry Res\_A(A: out Vector\_6H);

143. end Task4;

144.

145. task Task5 is

146. entry Get\_ai(mi: out Integer);

147. entry Set\_a(m: in Integer);

148. end Task5;

149.

150. task Task6 is

151. entry SetData(B: in Vector\_H; MK: in Matrix\_H;

152. Z: in Vector\_H; E: in Vector\_N; MO: in Matrix\_N);

153.

154. entry Get\_ai(mi: out Integer);

155. entry Set\_a(m: in Integer);

156.

157. entry Res\_A(A: out Vector\_H);

158. end Task6;

159.

160. task Task7 is

161. entry SetData(B: in Vector\_H; MK: in Matrix\_H;

162. Z: in Vector\_H; E: in Vector\_N; MO: in Matrix\_N);

163.

164. entry Get\_ai(mi: out Integer);

165. entry Set\_a(m: in Integer);

166.

167. entry Res\_A(A: out Vector\_H);

168. end Task7;

169.

170. --------------------------------------------------------------

171.

172. task body Task1 is

173. Z\_N: Vector\_N;

174. E\_N: Vector\_N;

175. MO\_N: Matrix\_N;

176.

177. B\_H: Vector\_H;

178. MK\_H: Matrix\_H;

179.

180. A\_H: Vector\_H;

181.

182. a1: Integer;

183. a: Integer;

184. begin

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

186.

187. VectorInput(Z\_N);

188. VectorInput(E\_N);

189. MatrixInput(MO\_N);

190.

191. accept GetData(Z: out Vector\_6H; E: out Vector\_N; MO: out Matrix\_N) do

192. Z := Z\_N(H+1..N);

193. E := E\_N;

194. MO := MO\_N;

195. end GetData;

196.

197. accept SetData(B: in Vector\_H; MK: in Matrix\_H) do

198. B\_H := B;

199. MK\_H := MK;

200. end SetData;

201.

202. a1 := Calc1(Z\_N(1..H));

203.

204. accept Get\_ai (mi : out Integer) do

205. mi := a1;

206. end Get\_ai;

207.

208. accept Set\_a (m : in Integer) do

209. a := m;

210. end Set\_a;

211.

212. Calc3(a, B\_H, E\_N, MO\_N, MK\_H, A\_H);

213.

214. accept Res\_A (A : out Vector\_H) do

215. A := A\_H;

216. end Res\_A;

217.

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

219. end Task1;

220.

221. --------------------------------------------------------------

222.

223. task body Task2 is

224. Z\_6H: Vector\_6H;

225. E\_N: Vector\_N;

226. MO\_N: Matrix\_N;

227.

228. B\_2H: Vector\_2H;

229. MK\_2H: Matrix\_2H;

230.

231. A\_2H: Vector\_2H;

232.

233. a1: Integer;

234. a2: Integer;

235. a: Integer;

236. begin

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

238.

239. Task1.GetData(Z\_6H, E\_N, MO\_N);

240.

241. accept GetData (Z : out Vector\_5H; E : out Vector\_N; MO : out Matrix\_N) do

242. Z:= Z\_6H(H+1..6\*H);

243. E:= E\_N;

244. MO:= MO\_N;

245. end GetData;

246.

247. accept SetData (B : in Vector\_2H; MK : in Matrix\_2H) do

248. B\_2H := B;

249. MK\_2H := MK;

250. end SetData;

251.

252. Task1.SetData(B\_2H(1..H), MK\_2H(1..H));

253.

254. a2 := Calc1(Z\_6H(1..H));

255.

256. Task1.Get\_ai(a1);

257.

258. a := Max(a1, a2);

259.

260. accept Get\_ai (mi : out Integer) do

261. mi := a;

262. end Get\_ai;

263.

264. accept Set\_a (m : in Integer) do

265. a := m;

266. end Set\_a;

267.

268. Task1.Set\_a(a);

269.

270. Calc3(a, B\_2H(H+1..2\*H), E\_N, MO\_N, MK\_2H(H+1..2\*H), A\_2H(H+1..2\*H));

271.

272. Task1.Res\_A(A\_2H(1..H));

273.

274. accept Res\_A (A : out Vector\_2H) do

275. A := A\_2H;

276. end Res\_A;

277.

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

279. end Task2;

280.

281. --------------------------------------------------------------

282.

283. task body Task3 is

284. Z\_5H: Vector\_5H;

285. E\_N: Vector\_N;

286. MO\_N: Matrix\_N;

287.

288. B\_4H: Vector\_4H;

289. MK\_4H: Matrix\_4H;

290.

291. A\_4H: Vector\_4H;

292.

293. a3: Integer;

294. a2: Integer;

295. a4: Integer;

296. a6: Integer;

297. a: Integer;

298. begin

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

300.

301. Task2.GetData(Z\_5H, E\_N, MO\_N);

302.

303. accept GetData (Z : out Vector\_3H; E : out Vector\_N; MO : out Matrix\_N) do

304. Z := Z\_5H(2\*H+1..5\*H);

305. E := E\_N;

306. MO := MO\_N;

307. end GetData;

308.

309. accept SetData (B : in Vector\_4H; MK : in Matrix\_4H) do

310. B\_4H := B;

311. MK\_4H := MK;

312. end SetData;

313.

314. Task2.SetData(B\_4H(1..2\*H), MK\_4H(1..2\*H));

315.

316. Task6.SetData(B\_4H(3\*H+1..4\*H), MK\_4H(3\*H+1..4\*H), Z\_5H(1..H), E\_N, MO\_N);

317.

318. a3 := Calc1(Z\_5H(H+1..2\*H));

319.

320. Task2.Get\_ai(a2);

321. a := Max(a3, a2);

322.

323. Task4.Get\_ai(a4);

324. a := Max(a, a4);

325.

326. Task6.Get\_ai(a6);

327. a := Max(a, a6);

328.

329. Task2.Set\_a(a);

330. Task4.Set\_a(a);

331. Task6.Set\_a(a);

332.

333. Calc3(a, B\_4H(3\*H+1..4\*H), E\_N, MO\_N, MK\_4H(3\*H+1..4\*H), A\_4H(3\*H+1..4\*H));

334.

335. Task2.Res\_A(A\_4H(1..2\*H));

336. Task6.Res\_A(A\_4H(2\*H+1..3\*H));

337.

338. accept Res\_A (A : out Vector\_4H) do

339. A := A\_4H;

340. end Res\_A;

341.

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

343. end Task3;

344.

345. --------------------------------------------------------------

346.

347. task body Task4 is

348. Z\_3H: Vector\_3H;

349. E\_N: Vector\_N;

350. MO\_N: Matrix\_N;

351.

352. B\_6H: Vector\_6H;

353. MK\_6H: Matrix\_6H;

354.

355. A\_6H: Vector\_6H;

356.

357. a4: Integer;

358. a5: Integer;

359. a7: Integer;

360. a: Integer;

361. begin

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

363.

364. accept SetData (B : in Vector\_6H; MK : in Matrix\_6H) do

365. B\_6H := B;

366. MK\_6H := MK;

367. end SetData;

368.

369. Task3.GetData(Z\_3H, E\_N, MO\_N);

370.

371. Task3.SetData(B\_6H(1..4\*H), MK\_6H(1..4\*H));

372.

373. accept GetData (Z : out Vector\_H; E : out Vector\_N; MO : out Matrix\_N) do

374. Z := Z\_3H(2\*H+1..3\*H);

375. E := E\_N;

376. MO := MO\_N;

377. end GetData;

378.

379. Task7.SetData(B\_6H(4\*H+1..5\*H), MK\_6H(4\*H+1..5\*H), Z\_3H(1..H), E\_N, MO\_N);

380.

381. a4 := Calc1(Z\_3H(H+1..2\*H));

382.

383. Task5.Get\_ai(a5);

384. a := Max(a4, a5);

385.

386. Task7.Get\_ai(a7);

387. a := Max(a, a7);

388.

389. accept Get\_ai (mi : out Integer) do

390. mi := a;

391. end Get\_ai;

392.

393. accept Set\_a (m : in Integer) do

394. a := m;

395. end Set\_a;

396.

397. Task5.Set\_a(a);

398. Task7.Set\_a(a);

399.

400. Calc3(a, B\_6H(5\*H+1..6\*H), E\_N, MO\_N, MK\_6H(5\*H+1..6\*H), A\_6H(5\*H+1..6\*H));

401.

402. Task3.Res\_A(A\_6H(1..4\*H));

403. Task7.Res\_A(A\_6H(4\*H+1..5\*H));

404.

405. accept Res\_A (A : out Vector\_6H) do

406. A := A\_6H;

407. end Res\_A;

408.

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

410. end Task4;

411.

412. --------------------------------------------------------------

413.

414. task body Task5 is

415. Z\_H: Vector\_H;

416. E\_N: Vector\_N;

417. MO\_N: Matrix\_N;

418.

419. B\_N: Vector\_N;

420. MK\_N: Matrix\_N;

421.

422. A\_N: Vector\_N;

423.

424. a5: Integer;

425. a: Integer;

426. begin

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

428.

429. VectorInput(B\_N);

430. MatrixInput(MK\_N);

431.

432. Task4.SetData(B\_N(1..6\*H), MK\_N(1..6\*H));

433. Task4.GetData(Z\_H, E\_N, MO\_N);

434.

435. a5 := Calc1(Z\_H);

436.

437. accept Get\_ai (mi : out Integer) do

438. mi := a5;

439. end Get\_ai;

440.

441. accept Set\_a (m : in Integer) do

442. a := m;

443. end Set\_a;

444.

445. Calc3(a, B\_N(6\*H+1..N), E\_N, MO\_N, MK\_N(6\*H+1..N), A\_N(6\*H+1..N));

446.

447. Task4.Res\_A(A\_N(1..6\*H));

448.

449. if N <= 14 then

450. VectorOutput(A\_N);

451. end if;

452.

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

454. end Task5;

455.

456. --------------------------------------------------------------

457.

458. task body Task6 is

459. Z\_H: Vector\_H;

460. E\_N: Vector\_N;

461. MO\_N: Matrix\_N;

462.

463. B\_H: Vector\_H;

464. MK\_H: Matrix\_H;

465.

466. A\_H: Vector\_H;

467.

468. a6: Integer;

469. a: Integer;

470. begin

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

472.

473. accept SetData (B : in Vector\_H; MK : in Matrix\_H; Z : in Vector\_H; E : in Vector\_N; MO : in Matrix\_N) do

474. B\_H := B;

475. MK\_H := MK;

476. Z\_H := Z;

477. E\_N := E;

478. MO\_N := MO;

479. end SetData;

480.

481. a6 := Calc1(Z\_H);

482.

483. accept Get\_ai (mi : out Integer) do

484. mi := a6;

485. end Get\_ai;

486.

487. accept Set\_a (m : in Integer) do

488. a := m;

489. end Set\_a;

490.

491. Calc3(a, B\_H, E\_N, MO\_N, MK\_H, A\_H);

492.

493. accept Res\_A (A : out Vector\_H) do

494. A := A\_H;

495. end Res\_A;

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

497. end Task6;

498.

499. --------------------------------------------------------------

500.

501. task body Task7 is

502. Z\_H: Vector\_H;

503. E\_N: Vector\_N;

504. MO\_N: Matrix\_N;

505.

506. B\_H: Vector\_H;

507. MK\_H: Matrix\_H;

508.

509. A\_H: Vector\_H;

510.

511. a6: Integer;

512. a: Integer;

513. begin

514. Put\_Line("Task 7 started...");

515.

516. accept SetData (B : in Vector\_H; MK : in Matrix\_H; Z : in Vector\_H; E : in Vector\_N; MO : in Matrix\_N) do

517. B\_H := B;

518. MK\_H := MK;

519. Z\_H := Z;

520. E\_N := E;

521. MO\_N := MO;

522. end SetData;

523.

524. a6 := Calc1(Z\_H);

525.

526. accept Get\_ai (mi : out Integer) do

527. mi := a6;

528. end Get\_ai;

529.

530. accept Set\_a (m : in Integer) do

531. a := m;

532. end Set\_a;

533.

534. Calc3(a, B\_H, E\_N, MO\_N, MK\_H, A\_H);

535.

536. accept Res\_A (A : out Vector\_H) do

537. A := A\_H;

538. end Res\_A;

539.

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

541. end Task7;

542.

543. --------------------------------------------------------------

544.

545. begin

546. null;

547. end Main;

547 lines: No errors