

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА №2**

**З ДИСЦИПЛІНИ “** **ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ”**

**НА ТЕМУ: “** **Семафори, мютекси, події, критичні секції у C#”**

**Виконав:**

Студент ІІІ курсу ФІОТ

групи ІО-82

Шендріков Євгеній

Номер у списку - 24

**Перевірив:**

Доцент Корочкін О. В.

м. Київ – 2021 р.

**Технічне завдання**

1. Розробити паралельний алгоритм рішення математичної задачі

на мові C#;

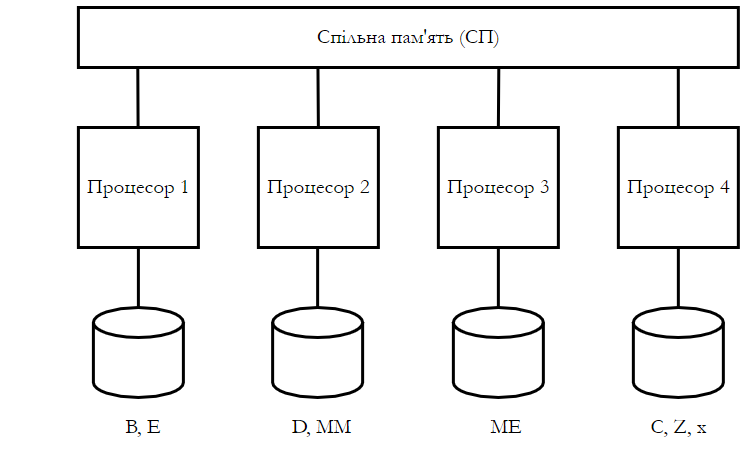
1. Виявити спільні ресурси;
2. Описати алгоритм кожного потоку (Т1 – Тр) з визначенням критичних ділянок (КД) і точок синхронізації (Wij , Sij);
3. Розробити структурну схему взаємодії задач, де застосувати всі вказані засоби взаємодії процесів;
4. Розробити програму (обов’язкові “шапка”, коментарі);
5. Виконати налагодження програми;
6. Отримати правильні результати обчислень;
7. За допомогою Диспетчеру задач Windows проконтролювати завантаження ядер процесору.

Задача: ;

Мова програмування: C#;

Засоби організації взаємодії: семафори, мютекси, події, критичні секції, атомік змінні (типи);

Структурна схема ПКС



**Виконання роботи**

**Етап 1. Побудова паралельного алгоритму**

1. bi = BH \* CH , i =
2. b = b + bi , i =
3. KH = sort(D \* (ME \* MMH))
4. K2H = mergesort(KH, KH)
5. K = mergesort(K2H, K2H)
6. ZH= KH + b \* EH \* x

Спільний ресурс: b, D, ME, x

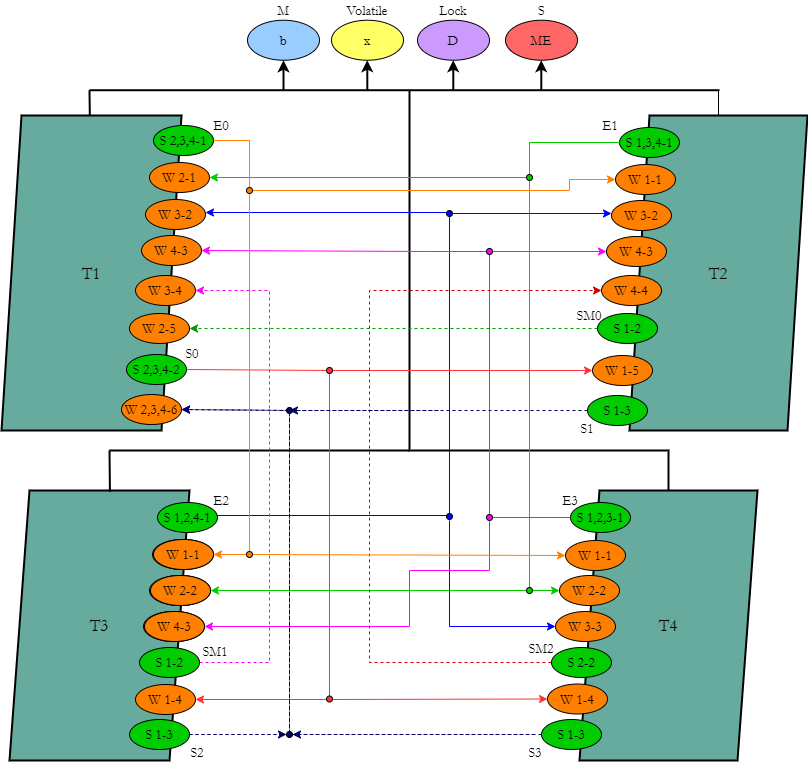
**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |  |
| --- | --- | --- |
| Задача Т1 | | ТС та КД |
| 1 | Введення B, E |  |
| 2 | Сигнал задачам Т2, Т3, Т4 про введення B, E |  |
| 3 | Чекати на введення D, MM у задачі Т2 |  |
| 4 | Чекати на введення ME у задачі Т3 |  |
| 5 | Чекати на введення C, Z, x у задачі Т4 |  |
| 6 | Копіювати x1 = x | КД |
| 7 | Копіювати ME1 = ME | КД |
| 8 | Копіювати D1 = D | КД |
| 9 | Обчислення b1 = BH \* CH |  |
| 10 | Обчислення b = b + b1 | КД |
| 11 | Обчислення KH = sort(D1 \* (ME1 \* MMH)) |  |
| 12 | Чекати на завершення обчислень KH у Т3 |  |
| 13 | Обчислення K2H = mergesort(KH, KH) |  |
| 14 | Чекати на завершення обчислень K2H у Т2 |  |
| 15 | Обчислення K = mergesort(K2H, K2H) |  |
| 16 | Сигнал задачам Т2, Т3, Т4 про обчислення K |  |
| 17 | Копіювати b1 = b | КД |
| 18 | Обчислення ZH= KH + b1 \* EH \* x1 |  |
| 19 | Чекати на завершення обчислень ZH в Т2, T3, T4 |  |
| 20 | Виведення результату Z |  |
|  | | |
| Задача Т2 | | ТС та КД |
| 1 | Введення D, MM |  |
| 2 | Сигнал задачам Т1, Т3, Т4 про введення D, MM |  |
| 3 | Чекати на введення B, E у задачі Т1 |  |
| 4 | Чекати на введення ME у задачі Т3 |  |
| 5 | Чекати на введення C, Z, x у задачі Т4 |  |
| 6 | Копіювати x2 = x | КД |
| 7 | Копіювати ME2 = ME | КД |
| 8 | Копіювати D2 = D | КД |
| 9 | Обчислення b2 = BH \* CH |  |
| 10 | Обчислення b = b + b2 | КД |
| 11 | Обчислення KH = sort(D2 \* (ME2 \* MMH)) |  |
| 12 | Чекати на завершення обчислень KH у Т4 |  |
| 13 | Обчислення K2H = mergesort(KH, KH) |  |
| 14 | Сигнал задачі Т1 про обчислення K2H |  |
| 15 | Чекати на завершення обчислень K у Т1 |  |
| 16 | Копіювати b2 = b | КД |
| 17 | Обчислення ZH = KH + b2 \* EH \* x2 |  |
| 18 | Сигнал задачі Т1 про обчислення ZH |  |
|  | | |
| Задача Т3 | | ТС та КД |
| 1 | Введення ME |  |
| 2 | Сигнал задачам Т1, Т2, Т4 про введення ME |  |
| 3 | Чекати на введення B, E у задачі Т1 |  |
| 4 | Чекати на введення D, MM у задачі Т2 |  |
| 5 | Чекати на введення C, Z, x у задачі Т4 |  |
| 6 | Копіювати x3 = x | КД |
| 7 | Копіювати ME3 = ME | КД |
| 8 | Копіювати D3 = D | КД |
| 9 | Обчислення b3 = BH \* CH |  |
| 10 | Обчислення b = b + b3 | КД |
| 11 | Обчислення KH = sort(D3 \* (ME3 \* MMH)) |  |
| 12 | Сигнал задачі Т1 про обчислення KH |  |
| 13 | Чекати на завершення обчислень K у Т1 |  |
| 14 | Копіювати b3 = b | КД |
| 15 | Обчислення ZH = KH + b3 \* EH \* x3 |  |
| 16 | Сигнал задачі Т1 про обчислення ZH |  |
|  | | |
| Задача Т4 | | ТС та КД |
| 1 | Введення C, x |  |
| 2 | Сигнал задачам Т1, Т2, Т3 про введення C, x |  |
| 3 | Чекати на введення B, E у задачі Т1 |  |
| 4 | Чекати на введення D, MM у задачі Т2 |  |
| 5 | Чекати на введення ME у задачі Т3 |  |
| 6 | Копіювати x4 = x | КД |
| 7 | Копіювати ME4 = ME | КД |
| 8 | Копіювати D4 = D | КД |
| 9 | Обчислення b4 = BH \* CH |  |
| 10 | Обчислення b = b + b4 | КД |
| 11 | Обчислення KH = sort(D4 \* (ME4 \* MMH)) |  |
| 12 | Сигнал задачі Т2 про обчислення KH |  |
| 13 | Чекати на завершення обчислень K у Т1 |  |
| 14 | Копіювати b4 = b | КД |
| 15 | Обчислення ZH = KH + b4 \* EH \* x4 |  |
| 16 | Сигнал задачі Т1 про обчислення ZH |  |

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

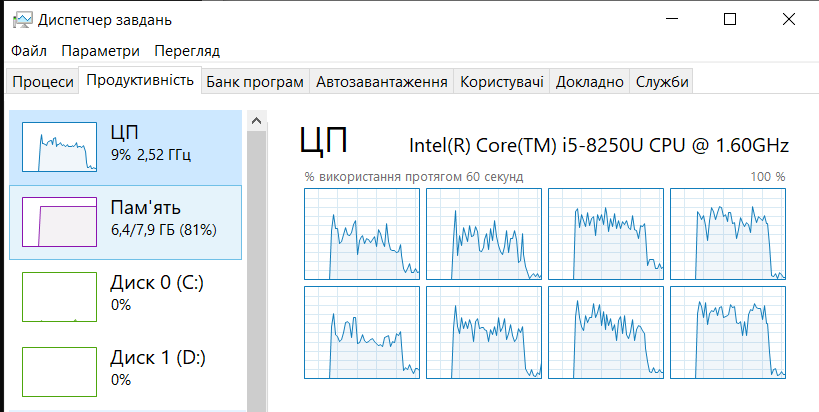
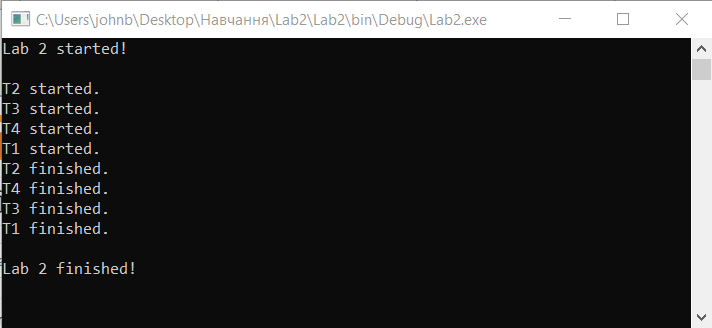
Умовні позначення на структурній схемі:

* *M* – мютекс для доступу до спільного ресурсу b;
* *volatile* – ключове слово для доступу до спільного ресурсу x;
* *Lock* – замок для доступу до спільного ресурсу D;
* *ME* – семафор для доступу до спільного ресурсу ME;
* *E0* – подія для синхронізації із завершенням вводу в T1;
* *E1* – подія для синхронізації із завершенням вводу в T2;
* *E2* – подія для синхронізації із завершенням вводу в T3;
* *E3* – подія для синхронізації із завершенням вводу в T4;
* S0 – семафор для синхронізації із завершенням злиття K у задачі T1;
* S1 – семафор для синхронізації завершення обчислень ZH у задачі T2;
* S2 – семафор для синхронізації завершення обчислень ZH у задачі T3;
* S3 – семафор для синхронізації завершення обчислень ZH у задачі T4;
* SM0 – семафор для синхронізації завершення злиття K2H у задачі Т2;
* SM1 – семафор для синхронізації завершення сортування KH у задачі Т3;
* SM2 – семафор для синхронізації завершення сортування KH у задачі Т4.



**Етап 4. Розробка програми**

**Результат роботи**

****

**Висновки**

1. На основі засобів мови С# було розроблено програму та паралельний алгоритм для рішення математичної задачі заданої за варіантом.
2. Було описано алгоритм кожного потоку (Т1 – Т4) з визначенням критичних ділянок (КД) та точок синхронізації (Wij , Sij);
3. Розроблено структурну схему взаємодії задач, де було застосовано всі вказані в завданні засоби взаємодії процесів. Для доступу до спільного ресурсу використовувався семафор, критична ділянки, volatile-змінна та мютекс, для синхронізації обчислень – бінарні семафори, а для синхронізації потоків – події.
4. Було перевірено працездатність програми, а також проконтрольовано завантаження ядер процесору за допомогою Диспетчеру задач. Програма забезпечує 80% завантаженості.

**Лістинг коду**

1. using System;

2. using System.Collections.Generic;

3. using System.Linq;

4. using System.Text;

5. using System.Threading;

6.

7. /\*-----------------------------------------------------

8. | Labwork #2 |

9. | PKS SP in C# |

10. ------------------------------------------------------

11. | Author | Jack (Yevhenii) Shendrikov |

12. | Group | IO-82 |

13. | Variant | #30 |

14. | Date | 23.02.2021 |

15. ------------------------------------------------------

16. | Function | Z = sort(D\*(ME\*MM)) + (B\*C)\*E\*x |

17. ------------------------------------------------------

18. \*/

19.

20. namespace Lab2

21. {

22. class Program : Operations

23. {

24. public const int N = 100;

25. public const int P = 4;

26. public const int H = N / P;

27.

28. public static Semaphore S0;

29. public static Semaphore S1;

30. public static Semaphore S2;

31. public static Semaphore S3;

32.

33. public static Semaphore SM0;

34. public static Semaphore SM1;

35. public static Semaphore SM2;

36.

37. public static EventWaitHandle E0;

38. public static EventWaitHandle E1;

39. public static EventWaitHandle E2;

40. public static EventWaitHandle E3;

41.

42. public static volatile int x;

43. public static Mutex mutex\_b = new Mutex(false);

44. public static object lockD = new object();

45. public static Semaphore S\_ME = new Semaphore(1, 1);

46.

47. public static int b = 0;

48.

49. public static Vector B = new Vector(N);

50. public static Vector C = new Vector(N);

51. public static Vector D = new Vector(N);

52. public static Vector E = new Vector(N);

53. public static Vector K = new Vector(N);

54. public static Vector Z = new Vector(N);

55.

56. public static Matrix ME = new Matrix(N);

57. public static Matrix MM = new Matrix(N);

58.

59.

60. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задача Т1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

61. public static void T1() {

62. Console.WriteLine("T1 started.");

63.

64. // 1 - Введення B, E

65. B = inputVector(N, 1);

66. E = inputVector(N, 1);

67.

68. // 2 - Сигнал задачам Т2, Т3, Т4 про введення B, E

69. E0.Set();

70.

71. // 3 - Чекати на введення D, MM у задачі Т2

72. E1.WaitOne();

73.

74. // 4 - Чекати на введення ME у задачі Т3

75. E2.WaitOne();

76.

77. // 5 - Чекати на введення C, Z, x у задачі Т4

78. E3.WaitOne();

79.

80. // 6 - Копіювати x1 = x

81. int x1 = x;

82.

83. // 7 - Копіювати ME1 = ME

84. Matrix ME1 = new Matrix(N);

85.

86. S\_ME.WaitOne();

87. ME1 = ME;

88. S\_ME.Release();

89.

90. // 8 - Копіювати D1 = D

91. Vector D1 = new Vector(N);

92. lock (lockD) {

93. D1 = D;

94. }

95.

96. // 9 - Обчислення b1 = BH \* CH

97. int b1 = mult(B, C, 0, H);

98.

99. // 10 - Обчислення b = b + b1

100. mutex\_b.WaitOne();

101. b += b1;

102. mutex\_b.ReleaseMutex();

103.

104. // 11 - Обчислення KH = sort(D1 \* (ME1 \* MMH))

105. K = mult(D1, mult(ME1, MM, 0, H), 0, H);

106. Vector buf = new Vector(H);

107. buf = sort(K, 0, H);

108. for (int i = 0; i < H; i++)

109. K.set(i, buf.get(i));

110.

111. // 12 - Чекати на завершення обчислень KH у Т3

112. SM1.WaitOne();

113.

114. // 13 - Злиття K2H = mergesort(KH, KH)

115. mergeSort(K, 0, 2 \* H);

116.

117. // 14 - Чекати на завершення обчислень K2H у Т2

118. SM0.WaitOne();

119.

120. // 15 - Обчислення K = mergesort(K2H, K2H)

121. mergeSort(K, 0, N);

122.

123. // 16 - Сигнал задачам Т2, Т3, Т4 про обчислення K

124. S0.Release();

125.

126. // 17 - Копіювати b1 = b

127. mutex\_b.WaitOne();

128. b1 = b;

129. mutex\_b.ReleaseMutex();

130.

131. // 18 - Обчислення ZH = KH + b1 \* EH \* x1

132. buf = add(K, mult(b1, mult(x1, E, 0, H), 0, H), 0, H);

133. for (int i = 0; i < H; i++)

134. Z.set(i, buf.get(i));

135.

136. // 19 - Чекати на завершення обчислень ZH в Т2, T3, T4

137. S1.WaitOne();

138. S2.WaitOne();

139. S3.WaitOne();

140.

141. // 20 - Виведення результату Z

142. outputVector(Z);

143.

144. Console.WriteLine("T1 finished.");

145. }

146.

147. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задача Т2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

148. public static void T2() {

149. Console.WriteLine("T2 started.");

150.

151. // 1 - Введення D, MM

152. D = inputVector(N, 1);

153. MM = inputMatrix(N, 1);

154.

155. // 2 - Сигнал задачам Т1, Т3, Т4 про введення D, MM

156. E1.Set();

157.

158. // 3 - Чекати на введення B, E у задачі Т1

159. E0.WaitOne();

160.

161. // 4 - Чекати на введення ME у задачі Т3

162. E2.WaitOne();

163.

164. // 5 - Чекати на введення C, Z, x у задачі Т4

165. E3.WaitOne();

166.

167. // 6 - Копіювати x2 = x

168. int x2 = x;

169.

170. // 7 - Копіювати ME2 = ME

171. Matrix ME2 = new Matrix(N);

172.

173. S\_ME.WaitOne();

174. ME2 = ME;

175. S\_ME.Release();

176.

177. // 8 - Копіювати D2 = D

178. Vector D2 = new Vector(N);

179. lock (lockD) {

180. D2 = D;

181. }

182.

183. // 9 - Обчислення b2 = BH \* CH

184. int b2 = mult(B, C, H, 2\*H);

185.

186. // 10 - Обчислення b = b + b2

187. mutex\_b.WaitOne();

188. b += b2;

189. mutex\_b.ReleaseMutex();

190.

191. // 11 - Обчислення KH = sort(D2 \* (ME2 \* MMH))

192. K = mult(D2, mult(ME2, MM, H, 2 \* H), H, 2 \* H);

193. Vector buf = new Vector(H);

194. buf = sort(K, H, 2 \* H);

195. for (int i = H; i < 2 \* H; i++)

196. K.set(i, buf.get(i));

197.

198. // 12 - Чекати на завершення обчислень KH у Т4

199. SM2.WaitOne();

200.

201. // 13 - Обчислення K2H = mergesort(KH, KH)

202. mergeSort(K, 2 \* H, N);

203.

204. // 14 - Сигнал задачі Т1 про обчислення K2H

205. SM0.Release();

206.

207. // 15 - Чекати на завершення обчислень K у Т1

208. S0.WaitOne();

209.

210. // 16 - Копіювати b2 = b

211. mutex\_b.WaitOne();

212. b2 = b;

213. mutex\_b.ReleaseMutex();

214.

215. // 17 - Обчислення ZH = KH + b2 \* EH \* x2

216. buf = add(K, mult(b2, mult(x2, E, H, 2 \* H), H, 2 \* H), H, 2 \* H);

217. for (int i = H; i < 2 \* H; i++)

218. Z.set(i, buf.get(i));

219.

220. // 18 - Сигнал задачі Т1 про обчислення ZH

221. S1.Release();

222.

223. Console.WriteLine("T2 finished.");

224. }

225.

226. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задача Т3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

227. public static void T3() {

228. Console.WriteLine("T3 started.");

229.

230. // 1 - Введення ME

231. ME = inputMatrix(N, 1);

232.

233. // 2 - Сигнал задачам Т1, Т2, Т4 про введення ME

234. E2.Set();

235.

236. // 3 - Чекати на введення B, E у задачі Т1

237. E0.WaitOne();

238.

239. // 4 - Чекати на введення D, MM у задачі Т2

240. E1.WaitOne();

241.

242. // 5 - Чекати на введення C, Z, x у задачі Т4

243. E3.WaitOne();

244.

245. // 6 - Копіювати x3 = x

246. int x3 = x;

247.

248. // 7 - Копіювати ME3 = ME

249. Matrix ME3 = new Matrix(N);

250.

251. S\_ME.WaitOne();

252. ME3 = ME;

253. S\_ME.Release();

254.

255. // 8 - Копіювати D3 = D

256. Vector D3 = new Vector(N);

257. lock (lockD) {

258. D3 = D;

259. }

260.

261. // 9 - Обчислення b3 = BH \* CH

262. int b3 = mult(B, C, 2 \* H, 3 \* H);

263.

264. // 10 - Обчислення b = b + b3

265. mutex\_b.WaitOne();

266. b += b3;

267. mutex\_b.ReleaseMutex();

268.

269. // 11 - Обчислення KH = sort(D3 \* (ME3 \* MMH))

270. K = mult(D3, mult(ME3, MM, 2 \* H, 3 \* H), 2 \* H, 3 \* H);

271. Vector buf = new Vector(H);

272. buf = sort(K, 2 \* H, 3 \* H);

273. for (int i = 2 \* H; i < 3 \* H; i++)

274. K.set(i, buf.get(i));

275.

276. // 12 - Сигнал задачі Т1 про обчислення KH

277. SM1.Release();

278.

279. // 13 - Чекати на завершення обчислень K у Т1

280. S0.WaitOne();

281.

282. // 14 - Копіювати b3 = b

283. mutex\_b.WaitOne();

284. b3 = b;

285. mutex\_b.ReleaseMutex();

286.

287. // 15 - Обчислення ZH = KH + b3 \* EH \* x3

288. buf = add(K, mult(b3, mult(x3, E, 2 \* H, 3 \* H), 2 \* H, 3 \* H), 2 \* H, 3 \* H);

289. for (int i = 2 \* H; i < 3 \* H; i++)

290. Z.set(i, buf.get(i));

291.

292. // 16 - Сигнал задачі Т1 про обчислення ZH

293. S2.Release();

294.

295. Console.WriteLine("T3 finished.");

296. }

297.

298. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задача Т4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

299. public static void T4() {

300. Console.WriteLine("T4 started.");

301. // 1 - Введення C, x

302. C = inputVector(N, 1);

303. x = 1;

304.

305. // 2 - Сигнал задачам Т1, Т2, Т4 про введення ME

306. E3.Set();

307.

308. // 3 - Чекати на введення B, E у задачі Т1

309. E0.WaitOne();

310.

311. // 4 - Чекати на введення D, MM у задачі Т2

312. E1.WaitOne();

313.

314. // 5 - Чекати на введення ME у задачі Т3

315. E2.WaitOne();

316.

317. // 6 - Копіювати x4 = x

318. int x4 = x;

319.

320. // 7 - Копіювати ME4 = ME

321. Matrix ME4 = new Matrix(N);

322.

323. S\_ME.WaitOne();

324. ME4 = ME;

325. S\_ME.Release();

326.

327. // 8 - Копіювати D4 = D

328. Vector D4 = new Vector(N);

329. lock (lockD) {

330. D4 = D;

331. }

332.

333. // 9 - Обчислення b4 = BH \* CH

334. int b4 = mult(B, C, 3 \* H, 4 \* H);

335.

336. // 10 - Обчислення b = b + b4

337. mutex\_b.WaitOne();

338. b += b4;

339. mutex\_b.ReleaseMutex();

340.

341. // 11 - Обчислення KH = sort(D4 \* (ME4 \* MMH))

342. K = mult(D4, mult(ME4, MM, 3 \* H, 4 \* H), 3 \* H, 4 \* H);

343. Vector buf = new Vector(H);

344. buf = sort(K, 3 \* H, 4 \* H);

345. for (int i = 3 \* H; i < 4 \* H; i++)

346. K.set(i, buf.get(i));

347.

348. // 12 - Сигнал задачі Т2 про обчислення KH

349. SM2.Release();

350.

351. // 13 - Чекати на завершення обчислень K у Т1

352. S0.WaitOne();

353.

354. // 14 - Копіювати b4 = b

355. mutex\_b.WaitOne();

356. b4 = b;

357. mutex\_b.ReleaseMutex();

358.

359. // 15 - Обчислення ZH = KH + b4 \* EH \* x4

360. buf = add(K, mult(b4, mult(x4, E, 3 \* H, 4 \* H), 3 \* H, 4 \* H), 3 \* H, 4 \* H);

361. for (int i = 3 \* H; i < 4 \* H; i++)

362. Z.set(i, buf.get(i));

363.

364. // 16 - Сигнал задачі Т1 про обчислення ZH

365. S3.Release();

366.

367. Console.WriteLine("T4 finished.");

368. }

369.

371. static void Main(string[] args) {

372. System.Console.WriteLine("Lab 2 started!\n");

373.

374. S0 = new Semaphore(0, 1);

375. S1 = new Semaphore(0, 1);

376. S2 = new Semaphore(0, 1);

377. S3 = new Semaphore(0, 1);

378.

379. SM0 = new Semaphore(0, 1);

380. SM1 = new Semaphore(0, 1);

381. SM2 = new Semaphore(0, 1);

382.

383. E0 = new EventWaitHandle(false, EventResetMode.ManualReset);

384. E1 = new EventWaitHandle(false, EventResetMode.ManualReset);

385. E2 = new EventWaitHandle(false, EventResetMode.ManualReset);

386. E3 = new EventWaitHandle(false, EventResetMode.ManualReset);

387.

388. Thread t1 = new Thread(T1);

389. Thread t2 = new Thread(T2);

390. Thread t3 = new Thread(T3);

391. Thread t4 = new Thread(T4);

392.

393. t1.Start();

394. t2.Start();

395. t3.Start();

396. t4.Start();

397. t1.Join();

398.

399. System.Console.WriteLine("\nLab 2 finished!\n");

400. Console.ReadKey();

401. }

402. }

403. }

Додаткові файли

**Operations.cs**

1. using System;

2. using System.Collections.Generic;

3. using System.Linq;

4. using System.Text;

5. using System.Threading.Tasks;

6. /\*-----------------------------------------------------

7. | Labwork #2 |

8. | PKS SP in C# |

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

10. | Author | Jack (Yevhenii) Shendrikov |

11. | Group | IO-82 |

12. | Variant | #30 |

13. | Date | 23.02.2021 |

14. ------------------------------------------------------

15. | Function | Z = sort(D\*(ME\*MM)) + (B\*C)\*E\*x |

16. ------------------------------------------------------

17. \*/

18.

19. namespace Lab2

20. {

21. class Operations

22. {

23.

24. public static Vector inputVector(int n, int value)

25. {

26. Vector vector = new Vector(n);

27. for (int i = 0; i < n; i++)

28. {

29. vector.set(i, value);

30. }

31. return vector;

32. }

33.

34. public static void outputVector(Vector vector)

35. {

36. if (vector.size() < 9)

37. {

38. Console.WriteLine(vector.toString());

39. }

40.

41. }

42.

43. public static Vector mult(Vector left, Matrix right, int l, int r)

44. {

45. Vector result = new Vector(left.size());

46. for (int i = l; i < r; i++)

47. {

48. result.set(i, 0);

49. for (int j = 0; j < left.size(); j++)

50. {

51. result.set(i, result.get(i) + left.get(j) \* right.get(j, i));

52. }

53. }

54. return result;

55. }

56.

57. public static Vector mult(int value, Vector vect, int l, int r)

58. {

59. Vector result = new Vector(vect.size());

60. for (int i = l; i < r; i++)

61. {

62. result.set(i, value \* vect.get(i));

63. }

64. return result;

65. }

66.

67. public static int mult(Vector vect1, Vector vect2, int l, int r)

68. {

69. int result = 0;

70. for (int i = l; i < r; i++)

71. result += vect1.get(i) \* vect2.get(i);

72. return result;

73. }

74.

75. public static Matrix inputMatrix(int n, int value)

76. {

77. Matrix matrix = new Matrix(n);

78. for (int i = 0; i < n; i++)

79. {

80. for (int j = 0; j < n; j++)

81. {

82. matrix.set(i, j, value);

83. }

84. }

85. return matrix;

86. }

87.

88. public static void outputMatrix(Matrix matrix)

89. {

90. if (matrix.size() < 9)

91. {

92. Console.WriteLine(matrix.toString());

93. }

94.

95. }

96.

97.

98. public static Matrix mult(Matrix left, Matrix right, int l, int r)

99. {

100. Matrix result = new Matrix(left.size());

101. for (int i = 0; i < left.size(); i++)

102. {

103. for (int j = l; j < r; j++)

104. {

105. result.set(i, j, 0);

106. for (int k = 0; k < left.size(); k++)

107. {

108. result.set(i, j, result.get(i, j) + left.get(i, k)

109. \* right.get(k, j));

110. }

111. }

112. }

113. return result;

114. }

115.

116. public static Vector add(Vector left, Vector right, int l, int r)

117. {

118. Vector result = new Vector(left.size());

119. for (int i = l; i < r; i++)

120. {

121. result.set(i, left.get(i) + right.get(i));

122. }

123. return result;

124. }

125.

126. public static Vector sort(Vector vector, int l, int r)

127. {

128. int tmp = vector.get(0);

129. Vector res;

130. res = vector;

131.

132. for (int i = l; i < r; i++)

133. {

134. for (int k = i + 1; k < r; k++)

135. {

136. if (res.get(i) > res.get(k))

137. {

138. tmp = res.get(k);

139. res.set(k, res.get(i));

140. res.set(i, tmp);

141. }

142. }

143. }

144.

145. return res;

146. }

147.

148. private static int[] merge(int[] left, int[] right)

149. {

150. int a = 0, b = 0;

151. int[] merged = new int[left.Length + right.Length];

152. for (Int32 i = 0; i < left.Length + right.Length; i++)

153. {

154. if (b < right.Length && a < left.Length)

155. if (left[a] > right[b] && b < right.Length)

156. merged[i] = right[b++];

157. else

158. merged[i] = left[a++];

159. else

160. if (b < right.Length)

161. merged[i] = right[b++];

162. else

163. merged[i] = left[a++];

164. }

165. return merged;

166. }

167.

168. public static void mergeSort(Vector vector, int l, int r)

169. {

170. if (vector.size() == 1)

171. return;

172. int mid = (r - l) / 2;

173. int[] merged = new int[r - l];

174. int[] array1 = new int[mid];

175. int[] array2 = new int[mid];

176. for (int i = 0; i < r - l; i++)

177. {

178. if (i < mid)

179. array1[i] = vector.get(i + l);

180. else

181. array2[i - mid] = vector.get(i + l);

182. }

183. merged = merge(array1, array2);

184. for (int i = l; i < r; i++)

185. vector.set(i, merged[i - l]);

186. }

187.

188.

189. }

190. }

191.

**Matrix.cs**

1. using System;

2. using System.Collections.Generic;

3. using System.Linq;

4. using System.Text;

5. using System.Threading.Tasks;

6.

7. /\*-----------------------------------------------------

8. | Labwork #2 |

9. | PKS SP in C# |

10. ------------------------------------------------------

11. | Author | Jack (Yevhenii) Shendrikov |

12. | Group | IO-82 |

13. | Variant | #30 |

14. | Date | 23.02.2021 |

15. ------------------------------------------------------

16. | Function | Z = sort(D\*(ME\*MM)) + (B\*C)\*E\*x |

17. ------------------------------------------------------

18. \*/

19.

20. namespace Lab2

21. {

22. class Matrix

23. {

24. private Vector[] vector;

25.

26. public Matrix(int n)

27. {

28. vector = new Vector[n];

29. for (int i = 0; i < vector.Length; i++)

30. {

31. vector[i] = new Vector(n);

32. }

33. }

34.

35. public void set(int n, int m, int val)

36. {

37. vector[n].set(m, val);

38. }

39.

40. public int get(int n, int m)

41. {

42. return vector[n].get(m);

43. }

44.

45. public Vector get(int index)

46. {

47. return vector[index];

48. }

49.

50. public int size()

51. {

52. return vector.Length;

53. }

54.

55. public String toString()

56. {

57. String res = "";

58. for (int i = 0; i < vector.Length; i++)

59. {

60. res += vector[i].toString();

61. if (i != vector.Length - 1)

62. {

63. res += "\n";

64. }

65. }

66. return res;

67. }

68.

69.

70. }

71. }

72.

**Vector.cs**

1. using System;

2. using System.Collections.Generic;

3. using System.Linq;

4. using System.Text;

5. using System.Threading.Tasks;

6. /\*-----------------------------------------------------

7. | Labwork #2 |

8. | PKS SP in C# |

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

10. | Author | Jack (Yevhenii) Shendrikov |

11. | Group | IO-82 |

12. | Variant | #30 |

13. | Date | 23.02.2021 |

14. ------------------------------------------------------

15. | Function | Z = sort(D\*(ME\*MM)) + (B\*C)\*E\*x |

16. ------------------------------------------------------

17. \*/

18.

19. namespace Lab2

20. {

21. class Vector

22. {

23. private int[] array;

24.

25. public Vector(int n)

26. {

27. array = new int[n];

28. }

29.

30. public void set(int index, int value)

31. {

32. array[index] = value;

33. }

34.

35. public int get(int index)

36. {

37. return array[index];

38. }

39.

40. public int size()

41. {

42. return array.Length;

43. }

44.

45. public String toString()

46. {

47. String res = "";

48. for (int i = 0; i < array.Length; i++)

49. {

50. res += " " + array[i];

51. }

52. return res;

53. }

54.

55.

56. }

57. }

58.