**Додаток Є. Лістинг коду програми ПРГ1**

GNAT 4.6.4

Copyright 1992-2010, Free Software Foundation, Inc.

Compiling: main.adb (source file time stamp: 2016-05-14 19:37:18)

1. with Ada.Text\_IO, Ada.Integer\_Text\_IO,Ada.Float\_Text\_IO, Ada.Synchronous\_Task\_Control, Ada.Real\_Time, MatrixOperations;

2. use Ada.Text\_IO, Ada.Integer\_Text\_IO,Ada.Float\_Text\_IO, Ada.Synchronous\_Task\_Control, Ada.Real\_Time;

3.

4.

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

6. -- --

7. -- Програмування паралельних компьютерних систем --

8. -- Курсова робота. ПРГ1. Ада. Захищені модулі --

9. -- --

10. -- Файл: parallelCalcs.ada --

11. -- Завдання: --

12. -- MA = MB\*(MC+MO)\*a + min(Z)\*MK --

13. -- --

14. -- --

15. -- --

16. -- Автор: Кахерський Олег, група IП-31 --

17. -- Дата: 12.03.2016 --

18. -- --

19. ------------------------------------------------------------------

20.

21. procedure Main is

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

23. p: Integer; --кількість задач

24. h: Integer; --кількість рядків матриць та векторів на одну задачу

25. StartTime: Time; --початок обчислень

26. DiffTime : Time\_Span; --різниця між поточним часом(завершення обчислень) та початковим часом

27. procedure Start is

28. package MatrixOperationsN is new MatrixOperations(N);

29. use MatrixOperationsN;

30. MA,MO,MC,MK: Matrix;

31. Z: Vector;

32. Task type ThreadTask(TaskNumber: Integer);

33. protected ProtectedModule is

34. entry waitInput; --бар'єр для синхронізації вводу даних

35. entry waitMinZCalcFinish; --бар'єр для синхронізації обчислень min(Z)

36. entry readyForOuptut; --очікування сигналу від T2-Tp задачею Т1

37. function readAlpha return Integer; --захищене зчитування alpha

38. function readMinZ return Integer; --захищене зчитування min(Z)

39. function readMB return Matrix; --захищене зчитування readMB

40. procedure inputFinishSignal; --надсилання сигналу про завершення вводу даних

41. procedure finishCalcsSignal; --надсилання сигналу про завершення обчислень

42. procedure finishMinZCalcSignal; --надсилання сигналу про завершення обчислень min(z,zi)

43. procedure setMinZ(b: Integer); --встановлення значення minZ

44. procedure initMB(Buff: Matrix); --ініціалізація MB

45. procedure initAlpha(const: Integer);--ініціалізація alpha

46. private

47. inputFlag: integer := 0; --флаг для синхронізації вводу

48. minZCalcFlag : integer := 0; --флаг для синхронізації завершення обчислень min(Z)

49. finCalcFlag : integer := 0; --флаг для синхронізації завершення обчислень мат. виразу

50. --спільні ресурси

51. a, minZ: Integer;

52. MB : Matrix;

53. end ProtectedModule;

54.

55. protected body ProtectedModule is

56.

57. --розблокування задач при inputFlag = 2 (завершення вводу даних в T1, Tp)

58. entry waitInput when inputFlag = 2 is

59. begin

60. null;

61. end waitInput;

62.

63. --розблокування задач при minZCalcFlag = p (завершення обчислення min(Z))

64. entry waitMinZCalcFinish when minZCalcFlag = p is

65. begin

66. null;

67. end waitMinZCalcFinish;

68.

69. --розблокування Т1 при завершенні обчислень в задачах T2..Tp

70. entry readyForOuptut when finCalcFlag = p-1 is

71. begin

72. null;

73. end readyForOuptut;

74.

75. function readAlpha return Integer is

76. begin

77. return a;

78. end readAlpha;

79.

80. function readMinZ return Integer is

81. begin

82. return minZ;

83. end readMinZ;

84.

85. function readMB return Matrix is

86. begin

87. return MB;

88. end readMB;

89.

90. procedure inputFinishSignal is

91. begin

92. inputFlag := inputFlag + 1;

93. end inputFinishSignal;

94.

95. procedure finishCalcsSignal is

96. begin

97. finCalcFlag := finCalcFlag + 1;

98. end finishCalcsSignal;

99.

100. procedure finishMinZCalcSignal is

101. begin

102. minZCalcFlag := minZCalcFlag + 1;

103. end finishMinZCalcSignal;

104.

105. procedure setMinZ(b: in Integer) is

106. begin

107. minZ := b;

108. end setMinZ;

109.

110. procedure initAlpha(const: Integer) is

111. begin

112. a := const;

113. end initAlpha;

114.

115. procedure initMB(Buff: Matrix) is

116. begin

117. MB := Buff;

118. end initMB;

119. end ProtectedModule;

120.

121. procedure compareWithMinZ(const:Integer) is

122. begin

123. Put\_Line("Task is ready for calc minZ");

124. New\_line;

125. if ProtectedModule.readMinZ >= const then

126. Put\_Line("Task is changing minZ");

127. New\_line;

128. ProtectedModule.setMinZ(const);

129. end if;

130. end compareWithMinZ;

131.

132. procedure calcMatrixEquation(a,zi,startIndex,endIndex : Integer; MB: Matrix) is

133. Buf1: Matrix;

134. begin

135. Matrix\_Matrix\_Add(MC,MO,startIndex,endIndex,1,Buf1); --Buf1\_h = MOh+MCh

136. Matrix\_Matrix\_Multiply(Buf1, MB, startIndex, endIndex, a, MA); --MA\_h = Buf1\_h\*MB

137. Matrix\_Matrix\_Add(MA,MK,startIndex,endIndex,zi,MA);

138. end calcMatrixEquation;

139.

140. task body ThreadTask is

141. tid: Integer := TaskNumber;

142. ai, zi, startIndex, endIndex: Integer;

143. MBi : Matrix;

144. begin

145. Put\_Line("Task " & Integer'Image(tid) & " started");

146. New\_line;

147. startIndex := h\*(tid - 1) + 1;

148. endIndex := h\*tid;

149. if tid = 1 then

150. Put\_Line("Task "& Integer'Image(tid) & " is initing data");

151. New\_line;

152. Put\_Line("Task "& Integer'Image(tid) & " inited MB");

153. New\_line;

154. ProtectedModule.initMB(Input\_Matrix);

155. Put\_Line("Task "& Integer'Image(tid) & " inited MO");

156. New\_line;

157. MO := Input\_Matrix;

158. Put\_Line("Task "& Integer'Image(tid) & " inited A");

159. New\_line;

160. ProtectedModule.initAlpha(Input\_Constant);

161. --сигнал задачам о завершении ввода

162. Put\_Line("Task "& Integer'Image(tid) & " finished input and send Signal to All");

163. New\_line;

164. ProtectedModule.inputFinishSignal;

165. end if;

166. if tid = p then

167. endIndex := n;

168. Put\_Line("Task "& Integer'Image(tid) & " is initing data");

169. New\_line;

170. Put\_Line("Task "& Integer'Image(tid) & " inited MC");

171. New\_line;

172. MC := Input\_Matrix;

173. Put\_Line("Task "& Integer'Image(tid) & " inited MK");

174. New\_line;

175. MK := Input\_Matrix;

176. Put\_Line("Task "& Integer'Image(tid) & " inited Z");

177. New\_line;

178. Z := Input\_Vector;

179. ProtectedModule.setMinZ(100000000);

180. Put\_Line("Task "& Integer'Image(tid) & " finished input and send Signal to All");

181. New\_line;

182. ProtectedModule.inputFinishSignal;

183. else

184. Put\_Line("Task "& Integer'Image(tid) & " waits for inputting data in T1, Tp");

185. New\_line;

186. end if;

187. ProtectedModule.waitInput;

188. zi := Min(Z,startIndex, endIndex);

189. Put\_Line("Task "& Integer'Image(tid) & " finished calcs minZ and sent signal of finishing");

190. compareWithMinZ(zi);

191. Put\_Line("Task "& Integer'Image(tid) & " is copying shared resources");

192. New\_line;

193. --копіювання СР

194. ai := ProtectedModule.readAlpha;

195. zi := ProtectedModule.readMinZ;

196. MBi := ProtectedModule.readMB;

197. Put\_Line("Task "& Integer'Image(tid) & " is calculating matrix equation");

198. New\_line;

199. calcMatrixEquation(ai,zi,startIndex,endIndex,MBi);

200. if tid = 1 then

201. Put\_Line("Task "& Integer'Image(tid) & " is waiting for signal from all(finishing calc matrix equation)");

202. New\_line;

203. ProtectedModule.readyForOuptut;

204. DiffTime := Clock - StartTime;

205. Put\_Line("Calculating time = ");

206. Put(Float(To\_Duration(DiffTime)));

207. New\_line;

208. if n <= 8 then

209. Put\_Line("Result:");

210. New\_line;

211. Output\_Matrix(MA);

212. end if;

213. Get(H);

214. else

215. Put\_Line("Task "& Integer'Image(tid) & " is sending signal to T1 (finishing calc matrix equation)");

216. New\_line;

217. ProtectedModule.finishCalcsSignal;

218. end if;

219. Put\_Line("Task " & Integer'Image(tid) & " finished");

220. New\_line;

221. end ThreadTask;

222. type ThreadTaskPointer is access ThreadTask;

223. type TaskArray is array (1 .. p) of ThreadTaskPointer;

224. tArray: TaskArray;

225. begin

226. for I in 1 .. p loop

227. tArray(I) := new ThreadTask(I);

228. end loop;

229. end Start;

230.

231. begin

232. Put("Program started");

233. New\_line;

234. Put("Input N");

235. New\_line;

236. Get(n);

237. Put("Input P");

238. New\_line;

239. Get(p);

240. StartTime := Clock;

241. h := n/p;

242. Start;

243. end Main;

243 lines: No errors

GNAT 4.6.4

Copyright 1992-2010, Free Software Foundation, Inc.

Compiling: matrixoperations.adb (source file time stamp: 2016-03-19 18:55:56)

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

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

3. ------------------------------------------------------------------

4. -- File: matrixOperations.adb --

5. -- --

6. -- Автор: Олег Кахерський, група ІП-31 --

7. -- Дата: 12.03.2016 --

8. ------------------------------------------------------------------

9. package body MatrixOperations is

10.

11. procedure Output\_Matrix (MA : Matrix) is

12. begin

13. New\_Line;

14. for I in Index loop

15. for J in Index loop

16. Put(Item => MA(i)(j), Width => 5);

17. end loop;

18. New\_line;

19. end loop;

20. end Output\_Matrix;

21.

22. procedure Matrix\_Matrix\_Multiply

23. (Left : Matrix;

24. Right : Matrix;

25. Start\_Index: Integer;

26. End\_Index: Integer;

27. const: Integer;

28. Result\_Matrix: out Matrix) is

29. begin

30. for i in Start\_Index..End\_Index loop

31. for J in Index loop

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

33. for K in Index loop

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

35. end loop;

36. Result\_Matrix(I)(J) := Result\_Matrix(I)(J) \* const;

37. end loop;

38. end loop;

39. end Matrix\_Matrix\_Multiply;

40.

41. procedure Matrix\_Matrix\_Add

42. (Left: Matrix;

43. Right: Matrix;

44. Start\_Index: Integer;

45. End\_Index: Integer;

46. const: Integer;

47. Result\_Matrix: out Matrix) is

48. begin

49. for I in Start\_Index .. End\_Index loop

50. for J in Index loop

51. Result\_Matrix(I)(J) := Left(I)(J) + Right(I)(J)\*const;

52. end loop;

53. end loop;

54. end Matrix\_Matrix\_Add;

55.

56. function Min

57. (Vect: Vector;

58. Start\_Index: Integer;

59. End\_Index: Integer) return Integer is

60. min: Integer;

61. begin

62. min := Vect(Start\_Index);

63. for I in Start\_Index .. End\_Index loop

64. if min > Vect(I) then

65. min := Vect(I);

66. end if;

67. end loop;

68. return min;

69. end Min;

70.

71. function Input\_Vector return Vector is

72. V: Vector;

73. begin

74. for I in Index loop

75. V(I) := 1;

76. end loop;

77. return V;

78. end Input\_Vector;

79.

80. function Input\_Constant return Integer is

81. begin

82. return 1;

83. end Input\_Constant;

84.

85. function Input\_Matrix return Matrix is

86. Result\_Matrix: Matrix;

87. begin

88. for I in Index loop

89. for J in Index loop

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

91. end loop;

92. end loop;

93. return Result\_Matrix;

94. end Input\_Matrix;

95.

96. end MatrixOperations;

Compiling: matrixoperations.ads (source file time stamp: 2016-03-15 20:00:35)

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

2. -- File: matrixOperations.ads --

3. -- --

4. -- Автор: Олег Кахерський, група ІП-31 --

5. -- Дата: 12.03.2016 --

6. ------------------------------------------------------------------

7. generic

8. N: in Integer; --розмірність масивів та матриць

9. package MatrixOperations is

10. type Vector is private;

11. type Matrix is private;

12.

13. --Вивід матриці на екран

14. procedure Output\_Matrix (MA : Matrix);

15.

16. --Множення матриці на матрицю(діапазон рядків StartIndex..EndIndex),

17. --результат множення множиться на константу

18. procedure Matrix\_Matrix\_Multiply

19. (Left : Matrix;

20. Right : Matrix;

21. Start\_Index: Integer;

22. End\_Index: Integer;

23. const: Integer; --константа, на яку множиться результат множення

24. Result\_Matrix: out Matrix);

25.

26. --Додавання матриць, Right(I)(J) перед додаванням множиться на константу

27. procedure Matrix\_Matrix\_Add

28. (Left: Matrix;

29. Right: Matrix;

30. Start\_Index: Integer;

31. End\_Index: Integer;

32. const: Integer;

33. Result\_Matrix: out Matrix);

34.

35. --Знаходження мінімального значення в діапазоні StartIndex, EndIndex

36. function Min

37. (Vect: Vector;

38. Start\_Index: Integer;

39. End\_Index: Integer) return Integer;

40.

41. --Ввід вектора

42. function Input\_Vector return Vector;

43.

44. --Ввід константи

45. function Input\_Constant return Integer;

46.

47. --Ввід матриці

48. function Input\_Matrix return Matrix;

49.

50. private

51. subtype Index is Integer range 1..N;

52. type Vector is array (Index) of Integer;

53. type Matrix is array (Index) of Vector;

54. end MatrixOperations;

96 lines: No errors