Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Основи паралельного програмування

Лабораторна робота №1

**«Потоки в мові АДА»**

Виконав:

студент групи ІВ-82

Троценко Д.А.

Перевірив:

Доцент Корочкін О. В.

Київ

2020 р.

**Завдання:**

**F1: d = (A\*((B+C)\*(MA\*ME)))**

**F2: MF = MIN(MH)\*MK\*ML**

**F3: O = MAX(MP\*MR)\*V**

**Лістинг програми:**

GNAT Community 2020 (20200818-93)

Copyright 1992-2020, Free Software Foundation, Inc.

Compiling: lab\_1.adb

Source file time stamp: 2020-09-16 12:49:18

Compiled at: 2020-09-16 16:00:19

**Lab\_1.adb**

**1. ----------------Main programm------------------------**

**2. --Parallel programming basics.**

**3. --Labwork 1. Ada. Subprograms and packages**

**4. --Trotsenko Daniil**

**5. --IV-82**

**6. --17.09.2020**

**7. --Func1: d = (A\*((B+C)\*(MA\*ME)))**

**8. --Func2: MF = MIN(MH)\*MK\*ML**

**9. --Func3: O = MAX(MP\*MR)\*V**

**10. -----------------------------------------------------**

11.

12.

13. with Data, Ada.Integer\_Text\_IO, Ada.Text\_IO, System.Multiprocessors;

14. use Ada.Integer\_Text\_IO, Ada.Text\_IO, System.Multiprocessors;

15.

16. procedure Lab\_1 is

17. n : Integer := 4;

18. package data1 is new data(n);

19. use data1;

20.

21. procedure tasks is

22.

23. --Task T1--

24. task T1 is

25. pragma Priority(1);

26. pragma Storage\_Size(100000);

27. pragma CPU(1);

28. end;

29.

30. task body T1 is

31. A, B, C: Vector;

32. MA, ME : Matrix;

33. d : Integer;

34. begin

35. Put\_Line("task T1 started");

36. Vector\_Filling\_Ones(A);

37. Vector\_Filling\_Ones(B);

38. Vector\_Filling\_Ones(C);

39.

40. Matrix\_Filling\_Ones(MA);

41. Matrix\_Filling\_Ones(ME);

42.

43. d := Func1(A, B, C, MA, ME);

44. delay(2.0);

45.

46. if n < 7 then

47. Put\_Line("--Func1: d = (A\*((B+C)\*(MA\*ME)))--");

48. Put("F1 : d = ");

49. Put(d);

50. New\_Line;

51. New\_Line;

52. end if;

53.

54. Put\_Line("task T1 finished");

55. New\_Line;

56.

57. end T1;

58.

59.

60. --Task T2--

61. task T2 is

62. pragma Priority(3);

63. pragma Storage\_Size(100000);

64. pragma CPU(2);

65. end;

66.

67. task body T2 is

68. MF, MH, MK, ML : Matrix;

69. begin

70. Put\_Line("task T2 started");

71. Matrix\_Filling\_Ones(MH);

72. Matrix\_Filling\_Ones(MK);

73. Matrix\_Filling\_Ones(ML);

74.

75. MF := Func2(MH,MK, ML);

76. delay(4.0);

77.

78. if n < 7 then

79. Put\_Line("--Func2: MF = MIN(MH)\*MK\*ML--");

80. Put\_Line("F2 : MF = ");

81. Matrix\_Output(MF);

82.

83. New\_Line;

84. New\_Line;

85. end if;

86.

87. Put\_Line("task T2 finished");

88. New\_Line;

89. end T2;

90.

91. task T3 is

92. pragma Priority(10);

93. pragma Storage\_Size(100000);

94. pragma CPU(3);

95. end;

96.

97. task body T3 is

98. V, O : Vector;

99. MR, MP : Matrix;

100. begin

101. Put\_Line("task T3 started");

102. New\_Line;

103.

104. Vector\_Filling\_Ones(V);

105.

106. Matrix\_Filling\_Ones(MR);

107. Matrix\_Filling\_Ones(MP);

108.

109. O := Func3(V, MR, MP);

110. delay(5.0);

111.

112. if n < 7 then

113. Put\_Line("--Func3: O = MAX(MP\*MR)\*V--");

114. Put\_Line("F3 : O = ");

115. Vector\_Output(O);

116.

117. New\_Line;

118. New\_Line;

119. end if;

120.

121. Put\_Line("task T3 finished");

122. New\_Line;

123. end T3;

124.

125. begin

126. null;

127. end tasks;

128.

129. begin

130. tasks;

131. end Lab\_1;

131 lines: No errors

1. --------------Package Data, body---------------

2.

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

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

5.

6. package body Data is

7.

8.

9.

10. --Max in Matrix

11. function Matrix\_Max(A: in Matrix) return Integer is

12. Max\_m : Integer := -10000;

13. begin

14. for i in 1..n loop

15. for j in 1..n loop

16. if A(i)(j) > Max\_m then

17. Max\_m := A(i)(j);

18. end if;

19. end loop;

20. end loop;

21. return Max\_m;

22. end Matrix\_Max;

23.

24. --Min in Matrix

25. function Matrix\_Min(A: in Matrix) return Integer is

26. Min\_m : Integer := 10000;

27. begin

28. for i in 1..n loop

29. for j in 1..n loop

30. if A(i)(j) < Min\_m then

31. Min\_m := A(i)(j);

32. end if;

33. end loop;

34. end loop;

35. return Min\_m;

36. end Matrix\_Min;

37.

38. -- Read Vector

39. procedure Vector\_Input(A: out Vector) is

40. begin

41. for i in 1..n loop

42. Get(A(i));

43. end loop;

44. end Vector\_Input;

45.

46. -- Write Vector on Screen

47. procedure Vector\_Output(A: in Vector) is

48. begin

49. for i in 1..n loop

50. Put(A(i));

51. Put(" ");

52. end loop;

53. end Vector\_Output;

54.

55. -- Read Matrix

56. procedure Matrix\_Input(A: out Matrix) is

57. begin

58. for i in 1..n loop

59. for j in 1..n loop

60. Get(A(i)(j));

61. end loop;

62. end loop;

63. end Matrix\_Input;

64.

65. -- Write Matrix on Screen

66. procedure Matrix\_Output(A: in Matrix) is

67. begin

68. for i in 1..n loop

69. for j in 1..n loop

70. Put(A(i)(j));

71. Put(" ");

72. end loop;

73. Put\_Line(" ");

74. end loop;

75. end Matrix\_Output;

76.

77. -- Multiplication of matrces

78.

79. function Matrix\_Multiplication(A, B : in Matrix) return Matrix is

80. P: Matrix;

81. S: Integer;

82. begin

83. for k in 1..n loop

84. for i in 1..n loop

85. s := 0;

86. for j in 1..n loop

87. S := S + A(k)(j) \* B(j)(i);

88. P(k)(i) := s;

89. end loop;

90. end loop;

91. end loop;

92. return P;

93. end Matrix\_Multiplication;

94.

95. -- Multiplication of vector and Matrix

96. function Vector\_Matrix\_Multiplication(A: in Vector; B: in Matrix) return Vector is

97. P: Vector;

98. s: Integer;

99. begin

100. for i in 1..n loop

101. s := 0;

102. for j in 1..n loop

103. s := s + A(i)\*B(j)(i);

104. end loop;

105.

106. P(i) := s;

107. end loop;

108. return P;

109. end Vector\_Matrix\_Multiplication;

110.

111.

112. -- Sum of Vectors

113. function Vector\_Sum(A, B: in Vector) return Vector is

114. S: Vector;

115. begin

116. for i in 1..n loop

117. s(i) := A(i) + B(i);

118. end loop;

119. return s;

120. end Vector\_Sum;

121.

122. -- Multiplication of vectors

123. function Vector\_Multiplication(A, B: in Vector) return Integer is

124. s: Integer;

125. begin

126. s := 0;

127. for i in 1..n loop

128. s := s + A(i) \* B(i);

129. end loop;

130. return s;

131. end Vector\_Multiplication;

132.

133.

134.

135. -- Sorting of Vector

136. procedure Vector\_Sorting(A: in out Vector) is

137. S: Integer;

138. begin

139. for i in 1..n loop

140. for j in 1..n loop

141.

142. if A(i) > A(j) then

143. S := A(j);

144. A(j) := A(i);

145. A(i) := S;

146. end if;

147.

148. end loop;

149. end loop;

150.

151. end Vector\_Sorting;

152.

153. -- Sorting Matrix

154. procedure Matrix\_Sorting(A: in out Matrix) is

155. begin

156. for i in 1..n loop

157. Vector\_Sorting(A(i));

158. end loop;

159. end Matrix\_Sorting;

160.

161.

162. --Sum Matrices

163. function Sum\_Matrix(A, B: in Matrix) return Matrix is

164. C : Matrix;

165. begin

166. for i in 1..n loop

167. for j in 1..n loop

168. C(i)(j) := A(i)(j) + B(i)(j);

169. end loop;

170. end loop;

171. return C;

172. end Sum\_Matrix;

173.

174. --Sub Matrices

175. function Sub\_Matrix(A, B: in Matrix) return Matrix is

176. C : Matrix;

177. begin

178. for i in 1..n loop

179. for j in 1..n loop

180. C(i)(j) := A(i)(j) - B(i)(j);

181. end loop;

182. end loop;

183. return C;

184. end Sub\_Matrix;

185.

186.

187. -- Transposition of Matrix

188. procedure Matrix\_Transposition(A: in out Matrix) is

189. S: Integer;

190. begin

191. for i in 1..n loop

192. for j in 1..n loop

193. S := A(j)(i);

194. A(j)(i) := A(i)(j);

195. A(i)(j) := S;

196.

197. end loop;

198. end loop;

199. end Matrix\_Transposition;

200.

201. -- Filling matrix with ones

202. procedure Matrix\_Filling\_Ones(A: out Matrix) is

203. begin

204. for i in 1..n loop

205. for j in 1..n loop

206. A(i)(j) := 1;

207. end loop;

208. end loop;

209. end Matrix\_Filling\_Ones;

210.

211. -- Filling vector with ones

212. procedure Vector\_Filling\_Ones(A: out Vector) is

213. begin

214. for i in 1..n loop

215. A(i) := 1;

216. end loop;

217. end Vector\_Filling\_Ones;

218.

219.

220.

221. procedure Matrix\_Filling\_Number(A: out Matrix; i, j, number:Integer) is

222. begin

223. A(i)(j) := number;

224. end Matrix\_Filling\_Number;

225.

226.

227. procedure Vector\_Fillig\_Number(A: out Vector; i, number: Integer) is

228. begin

229. A(i) := number;

230. end Vector\_Fillig\_Number;

231.

232.

233.

234. -- Calculation function 1

235. function Func1 (A, B, C: in Vector; MA, ME: in Matrix) return Integer is

236. begin

237. return Vector\_Multiplication(A,

238. Vector\_Matrix\_Multiplication(

239. Vector\_Sum(B, c),

240. Matrix\_Multiplication(MA, ME)

241. ));

242. end Func1;

243.

244.

245. -- Constant Matrix multiplication

246. procedure Constant\_Matrix\_Multiplication(a : Integer; B : in out Matrix) is

247. begin

248. for i in 1..n loop

249. for j in 1..n loop

250. B(i)(j) := B(i)(j) \* a;

251. end loop;

252. end loop;

253. end Constant\_Matrix\_Multiplication;

254.

255. -- Constant Vector Multiplication

256. procedure Constant\_Vector\_Multiplication(a : Integer; B : in out Vector) is

257. begin

258. for i in 1..n loop

259. B(i) := B(i) \* a;

260. end loop;

261. end Constant\_Vector\_Multiplication;

262.

263. --Calculate function 2

264. function Func2 (MH, MK, ML: in Matrix) return Matrix is

265. MF : Matrix;

266. begin

267. MF := Matrix\_Multiplication(MK, ML);

268. Constant\_Matrix\_Multiplication(Matrix\_Min(MH), MF);

269. return MF;

270. end Func2;

271.

272. -- Calculate function 3

273. function Func3 (V : in out Vector; MP, MR : in Matrix) return Vector is

274. begin

275. Constant\_Vector\_Multiplication(Matrix\_Max(Matrix\_Multiplication(MP, MR)), V);

276. return V;

277. end Func3;

278. end Data;

Compiling: data.ads

Source file time stamp: 2020-09-17 09:05:06

Compiled at: 2020-09-17 12:09:17

1. generic

2. n: Integer;

3.

4. package Data is

5. --- Declaration of private types

6. type Vector is private;

7. type Matrix is private;

8.

9. -- Read Vector

10. procedure Vector\_Input(A: out Vector);

11.

12. -- Write Vector on screen

13. procedure Vector\_Output(A: in Vector);

14.

15. -- Read Matrix

16. procedure Matrix\_Input(A: out Matrix);

17.

18. -- Write Matrix on screen

19. procedure Matrix\_Output(A: in Matrix);

20.

21. -- Calculation functions

22. function Func1 (A, B, C: in Vector; MA, ME: in Matrix) return Integer;

23. function Func2 (MH, MK, ML: in Matrix) return Matrix;

24. function Func3 (V : in out Vector; MP, MR : in Matrix) return Vector;

25.

26. -- Filling matrix with ones

27. procedure Matrix\_Filling\_Ones(A: out Matrix);

28.

29. -- Filling vector with ones

30. procedure Vector\_Filling\_Ones(A: out Vector);

31.

32.

33. procedure Matrix\_Filling\_Number(A: out Matrix; i,j, number:Integer);

34.

35. procedure Vector\_Fillig\_Number(A: out Vector; i, number:Integer);

36.

37. -- Determination private types

38. private

39. type Vector is array(1..n) of Integer;

40. type Matrix is array(1..n) of Vector;

41.

42.

43.

44. end Data;

278 lines: No errors

**Результат програми:**

C:\Users\Daniil Trotsenko\Documents\Educate\OPP\obj\lab\_1.exe

task T1 started

task T2 started

task T3 started

--Func1: d = (A\*((B+C)\*(MA\*ME)))--

F1 : d = 128

task T1 finished

--Func2: MF = MIN(MH)\*MK\*ML--

F2 : MF =

4 4 4 4

4 4 4 4

4 4 4 4

4 4 4 4

task T2 finished

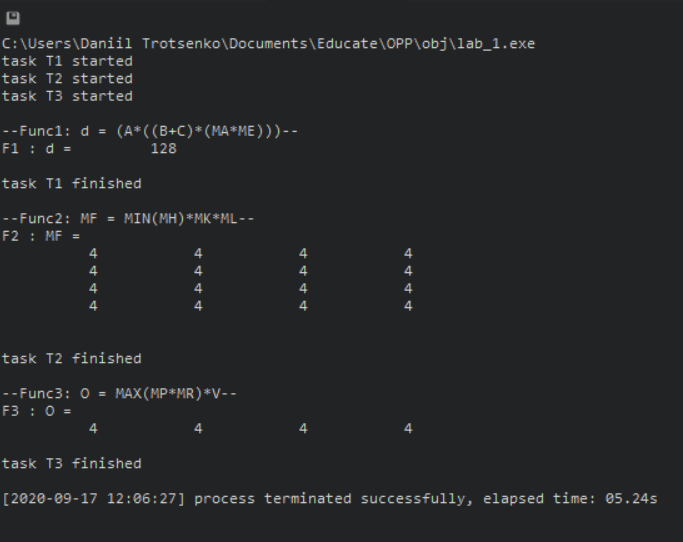
--Func3: O = MAX(MP\*MR)\*V--

F3 : O =

4 4 4 4

task T3 finished

[2020-09-17 12:06:27] process terminated successfully, elapsed time: 05.24s



**Висновки:**

Під час написання програми виникли деякі проблеми. **1-ша**. Якщо прибрати оператор **delay(n)** або зробити значення **n** рівним, то вивід кожної задачі в потоках буде перемішаним, іншими словами, значення матриць та векторів будуть перемішані. **2-га**. Якщо ставити різні пріорітети на задачі, можна помітити, що задачі T1-T3 з пріорітетами 1-3, можуть визиватися не поочередно. З пріорітетами для задачі T1, T2, T3 відповідно 1, 3, 5 я отримав такий запуск задач – T2, T3, T1. При збільшенні пріорітету, задачі почали виконуватися поочередно.**3-тя**. При сбробі організувати ввід з клавіатури, виникали проблеми з зчитуванням. Для уникнення цього, потрібно було б застосувати семафори з остановкою потоків, але в мене це коректно не вийшло. Для перевірки программи зробив автоматичне заповнення векторів та матриць 1-цями. Правильність роботи програми перевірив на простих прикладах розмірів n=2.