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:

Корочкін О. В.

Київ

2020 р.

**Завдання:**

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

F2: F = SORT(MG + TRANS(MH \* MK) - TRANS(ML))

F3: S = (O + P + V) \* (MR \* MS)

**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: e = ((A+B)\*(C+D\*(MA\*ME)))**

**8. --Func2: F = SORT(MG + TRANS(MH \* MK) - TRANS(ML))**

**9. --Func3: S = (O + P + V) \* (MR \* MS)**

**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 := 3;

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, D: Vector;

32. MA, ME : Matrix;

33. e : 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. Vector\_Filling\_Ones(D);

40.

41. Matrix\_Filling\_Ones(MA);

42. Matrix\_Filling\_Ones(ME);

43.

44. e := Func1(A, B, C, D, MA, ME);

45. delay(2.0);

46.

47. if n < 7 then

48. Put\_Line("--Func1: e = ((A+B)\*(C+D\*(MA\*ME)))--");

49. Put("F1 : e = ");

50. Put(e);

51. New\_Line;

52. New\_Line;

53. end if;

54.

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

56. New\_Line;

57.

58. end T1;

59.

60.

61. --Task T2--

62. task T2 is

63. pragma Priority(3);

64. pragma Storage\_Size(100000);

65. pragma CPU(2);

66. end;

67.

68. task body T2 is

69. MF, MG, MH, MK, ML : Matrix;

70. begin

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

72. Matrix\_Filling\_Ones(MG);

73. Matrix\_Filling\_Ones(MH);

74. Matrix\_Filling\_Ones(MK);

75. Matrix\_Filling\_Ones(ML);

76.

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

78. delay(4.0);

79.

80. if n < 7 then

81. Put\_Line("--Func2: F = SORT(MG + TRANS(MH \* MK) - TRANS(ML))--");

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

83. Matrix\_Output(MF);

84.

85. New\_Line;

86. New\_Line;

87. end if;

88.

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

90. New\_Line;

91. end T2;

92.

93. task T3 is

94. pragma Priority(10);

95. pragma Storage\_Size(100000);

96. pragma CPU(3);

97. end;

98.

99. task body T3 is

100. O, P, V, S : Vector;

101. MR, MS : Matrix;

102. begin

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

104. New\_Line;

105.

106. Vector\_Filling\_Ones(O);

107. Vector\_Filling\_Ones(P);

108. Vector\_Filling\_Ones(V);

109.

110. Matrix\_Filling\_Ones(MR);

111. Matrix\_Filling\_Ones(MS);

112.

113. S := Func3(O, P, V, MR, MS);

114. delay(5.0);

115.

116. if n < 7 then

117. Put\_Line("--Func3: S = (O + P + V) \* (MR \* MS)--");

118. Put\_Line("F3 : S = ");

119. Vector\_Output(S);

120.

121. New\_Line;

122. New\_Line;

123. end if;

124.

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

126. New\_Line;

127. end T3;

128.

129. begin

130. null;

131. end tasks;

132.

133. begin

134. tasks;

135. end Lab\_1;

135 lines: No errors

Compiling**: data.adb**

Source file time stamp: 2020-09-16 10:58:20

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

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. -- Read Vector

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

10. begin

11. for i in 1..n loop

12. Get(A(i));

13. end loop;

14. end Vector\_Input;

15.

16. -- Write Vector on Screen

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

18. begin

19. for i in 1..n loop

20. Put(A(i));

21. Put(" ");

22. end loop;

23. end Vector\_Output;

24.

25. -- Read Matrix

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

27. begin

28. for i in 1..n loop

29. for j in 1..n loop

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

31. end loop;

32. end loop;

33. end Matrix\_Input;

34.

35. -- Write Matrix on Screen

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

37. begin

38. for i in 1..n loop

39. for j in 1..n loop

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

41. Put(" ");

42. end loop;

43. Put\_Line(" ");

44. end loop;

45. end Matrix\_Output;

46.

47. -- Multiplication of matrces

48.

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

50. P: Matrix;

51. S: Integer;

52. begin

53. for k in 1..n loop

54. for i in 1..n loop

55. s := 0;

56. for j in 1..n loop

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

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

59. end loop;

60. end loop;

61. end loop;

62. return P;

63. end Matrix\_Multiplication;

64.

65. -- Multiplication of vector and Matrix

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

67. P: Vector;

68. s: Integer;

69. begin

70. for i in 1..n loop

71. s := 0;

72. for j in 1..n loop

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

74. end loop;

75.

76. P(i) := s;

77. end loop;

78. return P;

79. end Vector\_Matrix\_Multiplication;

80.

81.

82. -- Sum of Vectors

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

84. S: Vector;

85. begin

86. for i in 1..n loop

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

88. end loop;

89. return s;

90. end Vector\_Sum;

91.

92. -- Multiplication of vectors

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

94. s: Integer;

95. begin

96. s := 0;

97. for i in 1..n loop

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

99. end loop;

100. return s;

101. end Vector\_Multiplication;

102.

103.

104.

105. -- Sorting of Vector

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

107. S: Integer;

108. begin

109. for i in 1..n loop

110. for j in 1..n loop

111.

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

113. S := A(j);

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

115. A(i) := S;

116. end if;

117.

118. end loop;

119. end loop;

120.

121. end Vector\_Sorting;

122.

123. -- Sorting Matrix

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

125. begin

126. for i in 1..n loop

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

128. end loop;

129. end Matrix\_Sorting;

130.

131.

132. --Sum Matrices

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

134. C : Matrix;

135. begin

136. for i in 1..n loop

137. for j in 1..n loop

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

139. end loop;

140. end loop;

141. return C;

142. end Sum\_Matrix;

143.

144. --Sub Matrices

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

146. C : Matrix;

147. begin

148. for i in 1..n loop

149. for j in 1..n loop

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

151. end loop;

152. end loop;

153. return C;

154. end Sub\_Matrix;

155.

156.

157. -- Transposition of Matrix

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

159. S: Integer;

160. begin

161. for i in 1..n loop

162. for j in 1..n loop

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

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

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

166.

167. end loop;

168. end loop;

169. end Matrix\_Transposition;

170.

171. -- Filling matrix with ones

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

173. begin

174. for i in 1..n loop

175. for j in 1..n loop

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

177. end loop;

178. end loop;

179. end Matrix\_Filling\_Ones;

180.

181. -- Filling vector with ones

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

183. begin

184. for i in 1..n loop

185. A(i) := 1;

186. end loop;

187. end Vector\_Filling\_Ones;

188.

189.

190.

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

192. begin

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

194. end Matrix\_Filling\_Number;

195.

196.

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

198. begin

199. A(i) := number;

200. end Vector\_Fillig\_Number;

201.

202.

203.

204. -- Calculation functions

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

206. S : Vector;

207. begin

208. S := Vector\_Sum(C, Vector\_Matrix\_Multiplication(D,

209. Matrix\_Multiplication(MA, ME)));

210.

211. return Vector\_Multiplication(Vector\_Sum(A, B), S);

212. end Func1;

213.

214.

215.

216. function Func2 (MG, MH, MK: in Matrix; ML: in out Matrix) return Matrix is

217. MF : Matrix;

218. MT : Matrix;

219. MTT : Matrix;

220. begin

221. MTT := Matrix\_Multiplication(MH, MK);

222. Matrix\_Transposition(MTT);

223. MT := Sum\_Matrix(MG, MTT);

224.

225. Matrix\_Transposition(ML);

226. MF := Sub\_Matrix(MT, ML);

227. Matrix\_Sorting(MF);

228.

229. return MF;

230. end Func2;

231.

232.

233. function Func3 (O, P, V : in Vector; MR, MS : in Matrix) return Vector is

234. S, L : Vector;

235. MK : Matrix;

236. begin

237. L := Vector\_Sum(Vector\_Sum(O, P), V);

238. MK := Matrix\_Multiplication(MR, MS);

239. S := Vector\_Matrix\_Multiplication(L, MK);

240.

241. return S;

242. end Func3;

243. end Data;

Compiling: **data.ads**

Source file time stamp: 2020-09-16 10:58:42

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

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, D: in Vector; MA, ME: in Matrix) return Integer;

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

24. function Func3 (O, P, V : in Vector; MR, MS : 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;

243 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: e = ((A+B)\*(C+D\*(MA\*ME)))--

F1 : e = 60

task T1 finished

--Func2: F = SORT(MG + TRANS(MH \* MK) - TRANS(ML))--

F2 : MF =

3 3 3

3 3 3

3 3 3

task T2 finished

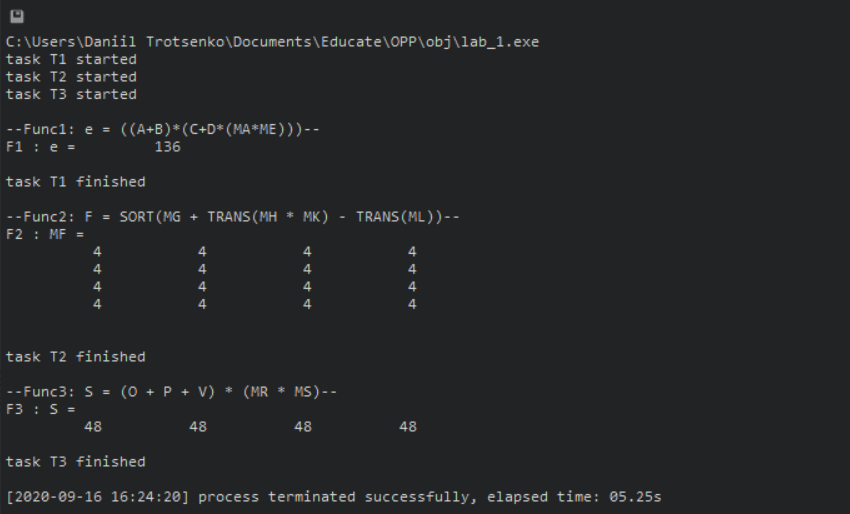
--Func3: S = (O + P + V) \* (MR \* MS)--

F3 : S =

27 27 27

task T3 finished

[2020-09-16 15:55:36] process terminated successfully, elapsed time: 05.21s



**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. Для перевірки программи зробив автоматичне заповнення векторів та матриць 1. Правильність роботи програми перевірив на простих прикладах розмірів n=2.