Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# Лабораторна робота № 2

з дисципліни «Паралельні і розподілені обчислення»

Виконав студент групи ІО-01 *Редько Олександр*

## Завдання



**Лістинг:**

Source file: ..\lab2.adb Wed Sep 17 20:51:58 2012

1 ------------------------------------------------------------------

2 -- --

3 -- Parallel and Distributed Computing --

4 -- Laboratory work #2. Tasks --

5 -- --

6 -- File: lab2.adb --

7 -- Task: F1: E = A + B + C + D \* (MA \* MZ) --

8 -- F2: MD = (MA \* MB) \* TRANS(MC) --

9 -- F3: E = (MA \* MM) \* B + MB \* SORT(A) --

10 -- --

11 -- Author: Redko Alexander, group IO-01 --

12 -- Date: 16.09.2012 --

13 -- --

14 ------------------------------------------------------------------

15

16 with Ada.Text\_IO, Ada.Integer\_text\_iO, Data;

17 use Ada.Text\_IO, Ada.Integer\_text\_iO;

18

19 procedure Lab2 is

20

21 N : Natural;

22 Fill : Character;

23 file1, File2, File3 : File\_Type;

24

25 begin

26

27 Put ("Input N: ");

28 Get (Item => N);

29

30

31 declare

32

33 package Data\_N is new Data(N);

34 use Data\_N;

35

36 Value : Integer := 0;

37

38 A1, A3, B1, B3, C1, D1, E1, E3 : Vector;

39 MA1, MA2, MA3, MB2, MB3, MC2, MD2, MM3, MZ1 : Matrix;

40

41 begin

42

43 Put ("Enter r (fill random), g (fill manually)"&

44 "or a (fill all the elements any number): ");

45 Get (Fill);

46 if (Fill = 'a') then

47 Put ("Enter number: ");

48 Get (Value);

49 end if;

50

51

52 Put\_Line ("Inputing data for F1...");

53 Input (Fill, Value, A1);

54 Input (Fill, Value, B1);

55 Input (Fill, Value, C1);

56 Input (Fill, Value, D1);

57 Input (Fill, Value, MA1);

58 Input (Fill, Value, MZ1);

59

60 Put\_Line ("Inputing data for F2...");

61 Input (Fill, Value, MA2);

62 Input (Fill, Value, MB2);

63 Input (Fill, Value, MC2);

64

65 Put\_Line ("Inputing data for F3...");

66 Input (fill, Value, MA3);

67 Input (Fill, Value, MM3);

68 Input (Fill, Value, B3);

69 Input (Fill, Value, MB3);

70 Input (Fill, Value, A3);

71

72 declare

73 task Task\_F1 is

74 pragma Priority(7);

75 end Task\_F1;

76

77 task body Task\_F1 is

78 begin

79 delay 0.0;

80 Put\_line("Started Task\_F1");

81 F1 (A1, B1, C1, D1, MA1, MZ1, E1);

82 Create (File1, Out\_File, "f1.txt");

83 Put\_Line (File1, "Result of F1:");

84 Output (E1, File1);

85 Close(File1);

86 Put\_line ("End Task\_F1");

87 end Task\_F1;

88

89 task Task\_F2 is

90 pragma Priority(7);

91 end Task\_F2;

92

93 task body Task\_F2 is

94 begin

95 delay 0.0;

96 Put\_line ("Started Task\_F2");

97 F2 (MA2, MB2, MC2, MD2);

98 Create (File2, Out\_File, "f2.txt");

99 Put\_Line (File2, "Result of F2:");

100 Output (MD2, File2);

101 Close(File2);

102 Put\_line ("End Task\_F2");

103 end Task\_F2;

104

105

106 task Task\_F3 is

107 pragma Priority(7);

108 end Task\_F3;

109

110 task body Task\_F3 is

111 begin

112 delay 0.0;

113 F3 (A3, B3, MA3, MB3, MM3, E3);

114 Create (File3, Out\_File, "f3.txt");

115 Put\_Line (File3, "Result of F3:");

116 Output (E3, File3);

117 Close(File3);

118 Put\_line ("End Task\_F3");

119 end Task\_F3;

120

121 begin

122 null;

123 end;

124

125

126 null;

127 end;

128

129 end Lab2;

130

131

132

Source file: ..\data.ads Wed Sep 17 20:51:58 2012

1 ------------------------------------------------------------------

2 -- --

3 -- Parallel and Distributed Computing --

4 -- Laboratory work #2. Tasks --

5 -- --

6 -- File: data.ads --

7 -- Task: F1: E = A + B + C + D \* (MA \* MZ) --

8 -- F2: MD = (MA \* MB) \* TRANS(MC) --

9 -- F3: E = (MA \* MM) \* B + MB \* SORT(A) --

10 -- --

11 -- Author: Redko Alexander, group IO-01 --

12 -- Date: 16.09.2012 --

13 -- --

14 ------------------------------------------------------------------

15

16 with Ada.Text\_IO;

17 use Ada.Text\_IO;

18

19 generic

20

21 N : in Natural; -- dimension of Vector and Matrix(N \* N)

22

23 package Data is

24

25 type Vector is private;

26 type Matrix is private;

27

28 procedure F1

29 (VA, VB, VC, VD : in Vector;

30 MA, MZ : in Matrix;

31 VE : out Vector);

32

33 procedure F2

34 (MA, MB, MC : in Matrix;

35 MD : out Matrix);

36

37 procedure F3

38 (VA, VB : in Vector;

39 MA, MB, MM : in Matrix;

40 VE : out Vector);

41

42 procedure Input (Way\_To\_Fill : in Character;

43 Value : in Integer;

44 V : out Vector);

45

46 procedure Input (Way\_To\_Fill : in Character;

47 Value : in Integer;

48 MA : out Matrix);

49

50 procedure Output (V : in Vector);

51

52 procedure Output (MA : in Matrix);

53

54 procedure Output (V : in Vector; File : File\_Type);

55

56 procedure Output (MA : in Matrix; File : File\_Type);

57

58

59 private

60

61 subtype Index is Positive range 1..N;

62 type Vector is array (Index) of integer;

63 type Matrix is array (Index) of Vector;

64

65

66 end Data;

Source file: ..\data.adb Wed Sep 17 20:51:58 2012

1 ------------------------------------------------------------------

2 -- --

3 -- Parallel and Distributed Computing --

4 -- Laboratory work #2. Tasks --

5 -- --

6 -- File: data.adb --

7 -- Task: F1: E = A + B + C + D \* (MA \* MZ) --

8 -- F2: MD = (MA \* MB) \* TRANS(MC) --

9 -- F3: E = (MA \* MM) \* B + MB \* SORT(A) --

10 -- --

11 -- Author: Redko Alexander, group IO-01 --

12 -- Date: 16.09.2012 --

13 -- --

14 ------------------------------------------------------------------

15

16 with Ada.Text\_IO, Ada.Integer\_Text\_IO, ada.numerics.discrete\_random, Ada.Strings, Ada.Strings.Fixed;

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

18

19 package body Data is

20

21 subtype Random\_Range is Integer range -20..20;

22 package Random\_Integer is new Ada.Numerics.Discrete\_Random(Random\_Range);

23 use Random\_Integer;

24 G : Random\_Integer.Generator;

25 Random\_Item : Random\_Range;

26

27

28 procedure Input (Way\_To\_Fill : in Character; Value : Integer; V : out Vector) is

29 Number : Integer;

30 begin

31 case Way\_To\_Fill is

32 when 'r' =>

33 Reset(G);

34 for I in Index loop

35 Random\_Item := Random(G);

36 V(I):= Random\_Item;

37 end loop;

38 when 'g' =>

39 Put ("Input vector: ");

40 for I in Index loop

41 Get(V(I));

42 end loop;

43 when 'a' =>

44 for I in Index loop

45 V(I) := Value;

46 end loop;

47 when others =>

48 for I in Index loop

49 V(I) := 1;

50 end loop;

51 end case;

52

53 end Input;

54

55

56 procedure Input (Way\_To\_Fill : in Character; Value : Integer; MA : out Matrix) is

57 Number : Integer := 3;

58 begin

59 case Way\_To\_Fill is

60 when 'r' =>

61 Reset(G);

62 for I in Index loop

63 for J in Index loop

64 Random\_Item := Random(G);

65 MA(I)(J) := Random\_Item;

66 end loop;

67 end loop;

68 when 'g' =>

69 Put ("Input matrix: ");

70 for I in Index loop

71 for J in Index loop

72 Get(MA(I)(J));

73 end loop;

74 end loop;

75 when 'a' =>

76 for I in Index loop

77 for J in Index loop

78 MA(I)(J) := Value;

79 end loop;

80 end loop;

81 when others =>

82 for I in Index loop

83 for J in Index loop

84 MA(I)(J) := 0;

85 end loop;

86 end loop;

87

88 end case;

89

90 end Input;

91

92

93 procedure Output (V : in Vector; File : File\_Type) is

94 begin

95 for I in Index loop

96 Put(File, V(I));

97 end loop;

98 New\_Line(File);

99 end Output;

100

101 procedure Output (MA : in Matrix; File : File\_Type) is

102 begin

103 for I in Index loop

104 for J in Index loop

105 Put(File, MA(I)(J));

106 end loop;

107 New\_line(File);

108 end loop;

109 New\_Line(File);

110 end Output;

111

112

113 procedure Output (V : in Vector) is

114 begin

115 New\_Line;

116 for I in Index loop

117 Put(Item => V(I), Width => 6);

118 end loop;

119 New\_Line;

120 end Output;

121

122

123 procedure Output (MA : in Matrix) is

124 begin

125 New\_Line;

126 for I in Index loop

127 for J in Index loop

128 Put(Item => MA(i)(j), Width => 6);

129 end loop;

130 New\_line;

131 end loop;

132 New\_Line;

133 end Output;

134

135

136 function "\*" --Matrix\_Matrix\_Multiply

137 (Left : Matrix;

138 Right : Matrix) return Matrix;

139

140

141 function "\*" --Vector\_Matrix\_Multiply

142 (Left : Vector;

143 Right : Matrix) return Vector;

144

145

146 function "+" --Vector\_Vector\_Add

147 (Left : Vector;

148 Right : Vector) return Vector;

149

150

151 procedure Transpose (A : in Matrix; R : out matrix);

152 procedure Sort (V : in Vector; A : out Vector);

153

154

155 procedure F1

156 (VA, VB, VC, VD : in Vector;

157 MA, MZ : in Matrix;

158 VE : out Vector) is

159 begin

160 VE := VA + VB + VC + VD \* (MA \* MZ);

161 end F1;

162

163

164 procedure F2

165 (MA, MB, MC : in Matrix;

166 MD : out Matrix) is

167 MCT : Matrix;

168 begin

169 Transpose(MC, MCT);

170 MD := (MA \* MB) \* MCT;

171

172 end F2;

173

174

175 procedure F3

176 (VA, VB : in Vector;

177 MA, MB, MM : in Matrix;

178 VE : out Vector) is

179 VS : Vector;

180 begin

181 Sort(VA, VS);

182 VE := VB \* (MA \* MM) + VS \* MB;

183 end F3;

184

185

186 function "\*"--Matrix\_Matrix\_Multiply

187 (Left : Matrix;

188 Right : Matrix) return Matrix

189 is

190 MR : Matrix;

191 begin

192 for i in Index loop

193 for J in Index loop

194 MR(I)(J) := 0;

195 for K in Index loop

196 MR(I)(J) := MR(I)(J) + Left(I)(K) \* Right(K)(J);

197 end loop;

198 end loop;

199 end loop;

200 return MR;

201 end "\*";

202

203

204 function "\*"--Vector\_Matrix\_Multiply

205 (Left : Vector;

206 Right : Matrix) return Vector

207 is

208 R : Vector;

209 begin

210 for J in Index loop

211 R(j) := 0;

212 begin

213 for K in Index loop

214 R(J) := R(J) + Left(K) \* Right(K )(J);

215 end loop;

216 end;

217 end loop;

218 return R;

219 end "\*";

220

221

222 function "+"--Vector\_Vector\_Add

223 (Left : Vector;

224 Right : Vector) return Vector

225 is

226 R : Vector;

227 begin

228 for J in Index loop

229 R (J) := Left (J) + Right (J);

230 end loop;

231 return R;

232 end "+";

233

234

235 procedure Transpose (A : in Matrix; R : out Matrix) is

236 begin

237 for J in Index loop

238 for K in Index loop

239 R (J)(K) := A (K)(j);

240 end loop;

241 end loop;

242 end Transpose;

243

244

245 procedure Sort (V : in Vector; A : out Vector) is

246 Min : Positive;

247 Temp : Integer;

248 begin

249 A := V;

250 for I in A'First..A'Last - 1 loop

251 Min := I;

252 for J in I + 1..A'Last loop

253 if A (Min) > A (J) then

254 Min := J;

255 end if;

256 end loop;

257 if Min /= I then

258 Temp := A (I);

259 A (I) := A (Min);

260 A (Min) := Temp;

261 end if;

262 end loop;

263 end Sort;

264

265 end Data;