НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №2**

з дисципліни **«**Паралельні та розподілені обчислення**»**

Виконав:

студент 3 курсу гр. ІО-31

Долинний Олександр

№ ЗК 3110

Перевірив:

Корочкін О. В.

Київ 2015 р.

***Завдання:***

1.1 3.21 3.11

F1: A = sort(B) \* (MB \* MC)

F2: W = sort(R \* MT) \* (MX \* MS)

F3: R = sort(S + T) \* trans(MS \* MR)

***Лістинг програми:***

GNAT GPL 2015 (20150428-49)

Copyright 1992-2015, Free Software Foundation, Inc.

Compiling: lab2.adb

Source file time stamp: 2015-10-06 06:59:28

Compiled at: 2015-10-06 10:02:44

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

2.

3. --Parallel and distributed computing.

4. --Labwork 2. Processes in Ada. Tasks

5. --Dolinniy Alexandr

6. --IO-31

7. --06.10.2015

8.

9. --Func1: A = sort(B)\*(MB\*MC)

10. --Func2: W = sort(R\*MT)\*(MX\*MS)

11. --Func3: R = sort(S+T)\*trans(MS\*MR)

12.

13.

14.

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

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

17. with data;

18.

19. procedure lab2 is

20.

21. CPU\_1 : CPU\_Range:=0;

22. CPU\_2 : CPU\_Range:=1;

23.

24. package dataNew is new data(3);

25. use dataNew;

26. --Specification of Task1 for Function1

27. task T1 is

28. pragma Task\_Name("Task 1");

29. pragma Priority(2);

30. pragma Storage\_Size(1000);

31. pragma CPU (CPU\_1);

32. end T1;

33. --Specification of Task2 for Function2

34. task T2 is

35. pragma Task\_Name("Task 2");

36. pragma Priority(4);

37. pragma Storage\_Size(1000);

38. pragma CPU (CPU\_2);

39. end T2;

40. --Specification of Task3 for Function3

41. task T3 is

42. pragma Task\_Name("Task 3");

43. pragma Priority(10);

44. pragma Storage\_Size(1000);

45. pragma CPU (CPU\_1);

46. end T3;

47.

48. --Body of Task1

49. --Func1: A = sort(B)\*(MB\*MC)

50. task body T1 is

51. MB, MC: matrix;

52. A, B: vector;

53. begin

54. Put\_Line("Task T1 started");

55. --Filling matrices and vectors with ones

56. Matrix\_Filling\_Ones(MB);

57. Matrix\_Filling\_Ones(MC);

58. Vector\_Filling\_Ones(B);

59. A := Func1(B, MB, MC);

60. --If the dimension of the vector A more than 10 result

61. -- not output

62. if Vector\_Length(A) <= 10 then

63. New\_Line;

64. Put("Func1: A = sort(B)\*(MB\*MC)");

65. New\_Line;

66. Vector\_Output(A);

67. end if;

68. New\_Line;

69. Put\_Line("Task T1 finished");

70. end T1;

71.

72. --Body of Task2

73. --Func2: W = sort(R\*MT)\*(MX\*MS)

74. task body T2 is

75. MT, MX, MS: Matrix;

76. W, R : Vector;

77. begin

78. Put\_Line("Task T2 started");

79. --Filling matrices and vectors with ones

80. Matrix\_Filling\_Ones(MT);

81. Matrix\_Filling\_Ones(MX);

82. Matrix\_Filling\_Ones(MS);

83. Vector\_Filling\_Ones(R);

84. delay 4.3;

85. --Calculation function

86. W := Func2(R, MT, MX, MS);

87. --If the dimension of the vector W more than 10 result

88. -- not output

89. if Vector\_Length(W) <= 10 then

90. New\_Line;

91. Put("Func2: W = sort(R\*MT)\*(MX\*MS)");

92. New\_Line;

93. Vector\_Output(W);

94. end if;

95. New\_Line;

96. Put\_Line("Task T2 finished");

97. end T2;

98.

99. --Body of Task3

100. --Func3: R = sort(S+T)\*trans(MS\*MR)

101. task body T3 is

102. MS, MR: matrix;

103. S, T, R: vector;

104. begin

105. Put\_Line("Task T3 started");

106. --Filling matrices and vectors with ones

107. Matrix\_Filling\_Ones(MS);

108. Matrix\_Filling\_Ones(MR);

109. Vector\_Filling\_Ones(S);

110. Vector\_Filling\_Ones(T);

111. delay 2.3;

112. --Calculation function

113. R := Func3(S, T, MS, MR);

114. --If the dimension of the vector R more than 10 result

115. -- not output

116. if Vector\_Length(R) <= 10 then

117. New\_Line;

118. Put("Func3: R = sort(S+T)\*trans(MS\*MR)");

119. New\_Line;

120. Vector\_Output(R);

121. end if;

122. New\_Line;

123. Put\_Line("Task T3 finished");

124. end T3;

125.

126. begin

127. null;

128. end lab2;

128 lines: No errors

GNAT GPL 2015 (20150428-49)

Copyright 1992-2015, Free Software Foundation, Inc.

Compiling: data.adb

Source file time stamp: 2015-10-06 06:42:44

Compiled at: 2015-10-06 10:01:58

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

2.

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

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

5. package body Data is

6.

7. --Read Vector

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

9. begin

10. for i in 1..n loop

11. Get(A(i));

12. end loop;

13. end Vector\_Input;

14.

15. --Write vector on screen

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

17. begin

18. for i in 1..n loop

19. Put(A(i));

20. Put(" ");

21. end loop;

22. end Vector\_Output;

23.

24. --Read matrix

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

26. begin

27. for i in 1..n loop

28. for j in 1..n loop

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

30. end loop;

31. end loop;

32. end Matrix\_Input;

33.

34. --Write matrix on screen

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

36. begin

37. for i in 1..n loop

38. for j in 1..n loop

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

40. Put(" ");

41. end loop;

42. Put\_Line(" ");

43. end loop;

44. end Matrix\_Output;

45.

46. --Multiplication of matrices

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

48. P: Matrix;

49. S: Integer;

50. begin

51. for k in 1..n loop

52. for i in 1..n loop

53. s := 0;

54. for j in 1..n loop

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

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

57. end loop;

58. end loop;

59. end loop;

60. return P;

61. end Matrix\_Multiplication;

62.

63. --Multiplication of vector and matrix

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

65. P: Vector;

66. s: Integer;

67. begin

68. for i in 1..n loop

69. s := 0;

70. for j in 1..n loop

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

72. end loop;

73. P(i) := S;

74. end loop;

75. return P;

76. end Vector\_Matrix\_Multiplication;

77.

78. --Sum of vectors

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

80. S: Vector;

81. begin

82. for i in 1..n loop

83. S(i) := A(i)+B(i);

84. end loop;

85. return S;

86. end Vector\_Sum;

87.

88. --Sorting of vector

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

90. S: Integer;

91. begin

92. for i in 1..n loop

93. for j in i..n loop

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

95. S:=A(j);

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

97. A(i):=S;

98. end if;

99. end loop;

100. end loop;

101. end Vector\_Sorting;

102.

103. --Transposition of Matrix

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

105. S: Integer;

106. begin

107. for i in 1..n loop

108. for j in i..n loop

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

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

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

112. end loop;

113. end loop;

114. end Matrix\_Transposition;

115.

116. --Filling matrix with ones

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

118. begin

119. for i in 1..n loop

120. for j in 1..n loop

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

122. end loop;

123. end loop;

124. end Matrix\_Filling\_Ones;

125.

126. --Filling vector with ones

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

128. begin

129. for i in 1..n loop

130. A(i) := 1;

131. end loop;

132. end Vector\_Filling\_Ones;

133.

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

135. begin

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

137. end Matrix\_Filling\_Number;

138.

139. procedure Vector\_Filling\_Number(A: out Vector; i,number:integer) is

140. begin

141. A(i) := number;

142. end Vector\_Filling\_Number;

143.

144. function Vector\_Length(A: in Vector) return Integer is

145. length: integer := 0;

146. begin

147. length := n;

148. return length;

149. end Vector\_Length;

150.

151.

152. --Calculation function 1

153. function Func1 (B: out Vector; MB, MC : in Matrix) return Vector is

154. MD:Matrix;

155. A:Vector;

156. begin

157. Vector\_Sorting(B);

158. MD:=Matrix\_Multiplication(MB,MC);

159. A:=Vector\_Matrix\_Multiplication(B,MD);

160. return A;

161. end Func1;

162.

163. --Calculation function 2

164. function Func2 (R: in Vector; MT, MX, MS : in Matrix) return Vector is

165. A,B:Vector;

166. MD: Matrix;

167. begin

168. A:=Vector\_Matrix\_Multiplication(R,MT);

169. Vector\_Sorting(A);

170. MD := Matrix\_Multiplication(MX, MS);

171. B:=Vector\_Matrix\_Multiplication(A,MD);

172. return B;

173. end Func2;

174.

175. --Calculation function 3

176. function Func3 (S,T: in Vector; MS, MR : in Matrix) return Vector is

177. R,A: Vector;

178. MD: Matrix;

179. begin

180. A := Vector\_Sum(S, T);

181. Vector\_Sorting(A);

182. MD := Matrix\_Multiplication(MS, MR);

183. Matrix\_Transposition(MD);

184. R:=Vector\_Matrix\_Multiplication(A,MD);

185. return R;

186. end Func3;

187. end Data;

Compiling: data.ads

Source file time stamp: 2015-10-06 06:42:44

Compiled at: 2015-10-06 10:01:58

1.

2. generic

3. n: Integer;

4. package data is

5.

6. ---Declaration of private types

7. type Vector is private;

8. type Matrix is private;

9.

10. --Read Vector

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

12.

13. --Write vector on screen

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

15.

16. --Read matrix

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

18.

19. --Write matrix on screen

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

21.

22. --Calculation function 1

23. function Func1 (B: out Vector; MB, MC : in Matrix) return Vector;

24.

25. --Calculation function 2

26. function Func2 (R: in Vector; MT, MX, MS : in Matrix) return Vector;

27.

28. --Calculation function 3

29. function Func3 (S,T: in Vector; MS, MR : in Matrix) return Vector;

30.

31.

32. --Filling matrix with ones

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

34.

35. --Filling vector with ones

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

37.

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

39.

40. procedure Vector\_Filling\_Number(A: out Vector; i,number:Integer);

41.

42. function Vector\_Length(A: in Vector) return Integer;

43.

44.

45. --Determination private types

46. private

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

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

49.

50. end Data;

187 lines: No errors