Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# Лабораторна робота № 6

з дисципліни «Паралельні і розподілені обчислення»

Виконав студент групи ІО-01 *Редько Олександр*

## Завдання



## Програма:

1 #include "stdafx.h"

2 #include **<**algorithm**>**

3 #include **<**iostream**>**

4 #include **<**omp**.**h**>**

5

6 **typedef** int**\*** vector**;**

7 **typedef** int**\*\*** matrix**;**

8

9 **using** **namespace** std**;**

10

11 /\*\*

12 \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

13 \* \*

14 \* Laboratory work #6. Threads with OpenMP \*

15 \* \*

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

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

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

19 \* \*

20 \* **@author** Red'ko Alexander \*

21 \* @group IO-01 \*

22 \* **@date** 19.10.12 \*

23 \* \*

24 \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

25 \*/

26

27 const int n **=** 5000**;**

28

29 vector inputVector**(**int**);**

30 matrix inputMatrix**(**int**);**

31 void outputVector**(**vector**);**

32 void outputMatrix**(**matrix**);**

33 void sortVector**(**vector**);**

34 vector add**(**vector**,** vector**);**

35 matrix mul**(**matrix**,** matrix**);**

36 vector mul**(**vector**,** matrix**);**

37 matrix transpose**(**matrix**);**

38 vector f1**(**vector**,** vector**,** vector**,** vector**,** matrix**,** matrix**);**

39 matrix f2**(**matrix**,** matrix**,** matrix**);**

40 vector f3**(**vector**,** vector**,** matrix**,** matrix**,** matrix**);**

41 void taskF1**(**void**);**

42 void taskF2**(**void**);**

43 void taskF3**(**void**);**

44

45 int \_tmain**(**int argc**,** \_TCHAR**\*** argv**[])**

46 **{**

47 cout **<<** "Lab6 started"**<<** endl**;**

48 #pragma comment**(**linker**,** "/STACK:50000000"**)**

49 #pragma omp parallel

50 **{**

51 #pragma omp sections nowait

52 **{**

53 #pragma omp section

54 **{**

55 taskF1**();**

56 **}**

57 #pragma omp section

58 **{**

59 taskF2**();**

60 **}**

61 #pragma omp section

62 **{**

63 taskF3**();**

64 **}**

65 **}**

66 **}**

67

68 cout **<<** "Lab6 ended"**<<** endl**;**

69 char ch**;**

70 cin **>>** ch**;**

71 **return** 0**;**

72 **}**

73

74 void taskF1**(**void**)**

75 **{**

76 cout **<<** "Task F1 started" **<<** endl**;**

77 vector a**,** b**,** c**,** d**,** e**;**

78 matrix ma**,** mz**;**

79 a **=** inputVector**(**1**);**

80 b **=** inputVector**(**1**);**

81 c **=** inputVector**(**1**);**

82 d **=** inputVector**(**1**);**

83 ma **=** inputMatrix**(**1**);**

84 mz **=** inputMatrix**(**1**);**

85 e **=** f1**(**a**,** b**,** c**,** d**,** ma**,** mz**);**

86 outputVector**(**e**);**

87 cout **<<** "Task F1 ended" **<<** endl**;**

88 **}**

89

90 void taskF2**(**void**)**

91 **{**

92 cout **<<** "Task F2 started" **<<** endl**;**

93 matrix ma**,** mb**,** mc**,** md**;**

94 ma **=** inputMatrix**(**1**);**

95 mb **=** inputMatrix**(**1**);**

96 mc **=** inputMatrix**(**1**);**

97 md **=** f2**(**ma**,** mb**,** mc**);**

98 outputMatrix**(**md**);**

99 cout **<<** "Task F2 ended" **<<** endl**;**

100 **}**

101

102 void taskF3**(**void**)**

103 **{**

104 cout **<<** "Task F3 started" **<<** endl**;**

105 vector a**,** b**,** e**;**

106 matrix ma**,** mb**,** mm**;**

107 a **=** inputVector**(**1**);**

108 b **=** inputVector**(**1**);**

109 ma **=** inputMatrix**(**1**);**

110 mb **=** inputMatrix**(**1**);**

111 mm **=** inputMatrix**(**1**);**

112 e **=** f3**(**a**,** b**,** ma**,** mb**,** mm**);**

113 outputVector**(**e**);**

114 cout **<<** "Task F3 ended" **<<** endl**;**

115 **}**

116

117 //E = A + B + C + D \* (MA \* MZ)

118 vector f1**(**vector a**,** vector b**,** vector c**,** vector d**,** matrix ma**,** matrix mz**)**

119 **{**

120 **return** add**(**add**(**add**(**a**,** b**),** c**),** mul**(**d**,** mul**(**ma**,** mz**)));**

121 **}**

122

123 //MD = (MA \* MB) \* TRANS(MC)

124 matrix f2**(**matrix ma**,** matrix mb**,** matrix mc**)**

125 **{**

126 **return** mul**(**mul**(**ma**,** mb**),** transpose**(**mc**));**

127 **}**

128

129 //E = B \* (MA \* MM) + SORT(A) \* MB

130 vector f3**(**vector a**,** vector b**,** matrix ma**,** matrix mb**,** matrix mm**)**

131 **{**

132 sortVector**(**a**);**

133 **return** add**(**mul**(**b**,** mul**(**ma**,** mm**)),** mul**(**a**,** mb**));**

134 **}**

135

136 vector inputVector**(**int value**)**

137 **{**

138 vector result **=** **new** int**[**n**];**

139 **for(**int i **=** 0**;** i **<** n**;** i**++)**

140 **{**

141 result**[**i**]** **=** value**;**

142 **}**

143 **return** result**;**

144 **}**

145

146 matrix inputMatrix**(**int value**)**

147 **{**

148 matrix result **=** **new** vector**[**n**];**

149 **for(**int i **=** 0**;** i **<** n**;** i**++)**

150 **{**

151 result**[**i**]** **=** **new** int**[**n**];**

152 **}**

153 **for(**int i **=** 0**;** i **<** n**;** i**++)**

154 **{**

155 **for(**int j **=** 0**;** j **<** n**;** j**++)**

156 **{**

157 result**[**i**][**j**]** **=** value**;**

158 **}**

159 **}**

160 **return** result**;**

161 **}**

162

163 void outputVector**(**vector v**)**

164 **{**

165 **if(**n **<=** 10**)**

166 **{**

167 **for(**int i **=** 0**;** i **<** n**;** i**++)**

168 **{**

169 cout **<<** v**[**i**]** **<<** " "**;**

170 **}**

171 cout **<<** endl**;**

172 **}**

173 **}**

174

175 void outputMatrix**(**matrix m**)**

176 **{**

177 **if(**n **<=** 10**)**

178 **{**

179 **for(**int i **=** 0**;** i **<** n**;** i**++)**

180 **{**

181 **for(**int j **=** 0**;** j **<** n**;** j**++)**

182 **{**

183 cout **<<** m**[**i**][**j**]** **<<** " "**;**

184 **}**

185 cout **<<** endl**;**

186 **}**

187 **}**

188 **}**

189

190 void sortVector**(**vector v**)**

191 **{**

192 sort**(**v**,** v **+** n**);**

193 **}**

194

195 vector add**(**vector left**,** vector right**)**

196 **{**

197 vector result **=** inputVector**(**0**);**

198 **for(**int i **=** 0**;** i **<** n**;** i**++)**

199 **{**

200 result**[**i**]** **=** left**[**i**]** **+** right**[**i**];**

201 **}**

202 **return** result**;**

203 **}**

204

205 matrix transpose**(**matrix m**)**

206 **{**

207 matrix result **=** inputMatrix**(**0**);**

208 **for(**int i **=** 0**;** i **<** n**;** i**++)**

209 **{**

210 **for(**int j **=** 0**;** j **<** n**;** j**++)**

211 **{**

212 result**[**i**][**j**]** **=** m**[**j**][**i**];**

213 **}**

214 **}**

215 **return** result**;**

216 **}**

218

219 matrix mul**(**matrix left**,** matrix right**)**

220 **{**

221 matrix result **=** inputMatrix**(**0**);**

222 **for(**int i **=** 0**;** i **<** n**;** i**++)**

223 **{**

224 **for(**int j **=** 0**;** j **<** n**;** j**++)**

225 **{**

226 result**[**i**][**j**]** **=** 0**;**

227 **for(**int k **=** 0**;** k **<** n**;** k**++)**

228 **{**

229 result**[**i**][**j**]** **+=** left**[**i**][**k**]** **\*** right**[**k**][**j**];**

230 **}**

231 **}**

232 **}**

233 **return** result**;**

234 **}**

235

236 vector mul**(**vector left**,** matrix right**)**

237 **{**

238 vector result **=** inputVector**(**0**);**

239 **for(**int j **=** 0**;** j **<** n**;** j**++)**

240 **{**

241 result**[**j**]** **=** 0**;**

242 **for(**int k **=** 0**;** k **<** n**;** k**++)**

243 **{**

244 result**[**j**]** **+=** left**[**k**]** **\*** right**[**k**][**j**];**

245 **}**

246 **}**

247 **return** result**;**

248 **}**