Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# Лабораторна робота № 5

з дисципліни «Паралельні і розподілені обчислення»

Виконав студент групи ІО-01 *Редько Олександр*

## Завдання



## Програма:

1 // pro\_lab5\_cpp.cpp : Defines the entry point for the console application.

2 //

3

4 #include "stdafx.h"

5 #include **<**iostream**>**

6 #include **<**algorithm**>**

7 #include **<**windows**.**h**>**

8

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

10

11 /\*\*

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

13 \* \*

14 \* Laboratory work #5. Threads in Win32 \*

15 \* \*

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

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

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

19 \* \*

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

21 \* @group IO-01 \*

22 \* **@date** 08.10.12 \*

23 \* \*

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

25 \*/

26

27 const int n **=** 1000**;**

28

29 void inputVector**(**int **[]);**

30 void inputMatrix**(**int **[][**n**]);**

31 void outputVector**(**int **[]);**

32 void outputMatrix**(**int **[][**n**]);**

33 void sortVector**(**int **[]);**

34 void add**(**int **[],** int **[],** int **[]);**

35 void mul**(**int **[][**n**],** int **[][**n**],** int **[][**n**]);**

36 void mul**(**int **[],** int **[][**n**],** int **[]);**

37 void transpose**(**int**[][**n**],** int**[][**n**]);**

38 void f1**(**int**[],** int**[],** int**[],** int**[],** int**[][**n**],** int**[][**n**],** int**[]);**

39 void f2**(**int**[][**n**],** int**[][**n**],** int**[][**n**],** int**[][**n**]);**

40 void f3**(**int**[],** int**[],** int**[][**n**],** int**[][**n**],** int **[][**n**],** int**[]);**

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

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

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

44

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

46 **{**

47 cout **<<** "Lab5 started"**<<** endl**;**

48 DWORD TidF1**,** TidF2**,** TidF3**;**

49 HANDLE hThreadF1**,** hThreadF2**,** hThreadF3**;**

50

51 hThreadF1 **=** CreateThread**(**

52 **NULL,** //атрибут безпеки

53 588065000**,** //розмір стека

54 **(**LPTHREAD\_START\_ROUTINE**)**taskF1**,** //функція потоку

55 **NULL,** //аргумент функції потоку

56 CREATE\_SUSPENDED**,** //прапорець

57 **&**TidF1 //ідентифікатор потоку

58 **);**

59 hThreadF2 **=** CreateThread**(NULL,** 588065000**,** **(**LPTHREAD\_START\_ROUTINE**)**taskF2**,**

60 **NULL,** CREATE\_SUSPENDED**,** **&**TidF2**);**

61 hThreadF3 **=** CreateThread**(NULL,** 588065000**,** **(**LPTHREAD\_START\_ROUTINE**)**taskF3**,**

62 **NULL,** CREATE\_SUSPENDED**,** **&**TidF3**);**

63

64 SetThreadPriority**(**hThreadF1**,** 3**);**

65 SetThreadPriority**(**hThreadF2**,** 3**);**

66 SetThreadPriority**(**hThreadF3**,** 3**);**

67

68 ResumeThread**(**hThreadF1**);**

69 ResumeThread**(**hThreadF2**);**

70 ResumeThread**(**hThreadF3**);**

71

72 CloseHandle**(**hThreadF1**);**

73 CloseHandle**(**hThreadF2**);**

74 CloseHandle**(**hThreadF3**);**

75

76 cout **<<** "Lab5 ended"**<<** endl**;**

77 char ch**;**

78 cin **>>** ch**;**

79 **return** 0**;**

80 **}**

81 //--------------------------------------------------------------

82 void taskF1**(**void**)**

83 **{**

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

85 int a**[**n**],** b**[**n**],** c**[**n**],** d**[**n**],** e**[**n**];**

86 int ma**[**n**][**n**],** mz**[**n**][**n**];**

87 inputVector**(**a**);**

88 inputVector**(**b**);**

89 inputVector**(**c**);**

90 inputVector**(**d**);**

91 inputMatrix**(**ma**);**

92 inputMatrix**(**mz**);**

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

94 outputVector**(**e**);**

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

96 **}**

97 //--------------------------------------------------------------

98 void taskF2**(**void**)**

99 **{**

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

101 int ma**[**n**][**n**],** mb**[**n**][**n**],** mc**[**n**][**n**],** md**[**n**][**n**];**

102 inputMatrix**(**ma**);**

103 inputMatrix**(**mb**);**

104 inputMatrix**(**mc**);**

105 f2**(**ma**,** mb**,** mc**,** md**);**

106 outputMatrix**(**md**);**

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

108 **}**

109 //--------------------------------------------------------------

110 void taskF3**(**void**)**

111 **{**

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

113 int a**[**n**],** b**[**n**],** e**[**n**];**

114 int ma**[**n**][**n**],** mb**[**n**][**n**],** mm**[**n**][**n**];**

115 inputVector**(**a**);**

116 inputVector**(**b**);**

117 inputMatrix**(**ma**);**

118 inputMatrix**(**mb**);**

119 inputMatrix**(**mm**);**

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

121 outputVector**(**e**);**

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

123 **}**

124 //--------------------------------------------------------------

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

126 void f1**(**int a**[],** int b**[],** int c**[],** int d**[],** int ma**[][**n**],** int mz**[][**n**],** int e**[])**

127 **{**

128 int ab**[**n**],** abc**[**n**],** dmamz**[**n**];**

129 int mamz**[**n**][**n**];**

130 add**(**a**,** b**,** ab**);**

131 add**(**ab**,** c**,** abc**);**

132 mul**(**ma**,** mz**,** mamz**);**

133 mul**(**d**,** mamz**,** dmamz**);**

134 add**(**abc**,** dmamz**,** e**);**

135 **}**

136

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

138 void f2**(**int ma**[][**n**],** int mb**[][**n**],** int mc**[][**n**],** int md**[][**n**])**

139 **{**

140 int mamb**[**n**][**n**],** tmc**[**n**][**n**];**

141 mul**(**ma**,** mb**,** mamb**);**

142 transpose**(**mc**,** tmc**);**

143 mul**(**mamb**,** tmc**,** md**);**

144 **}**

145

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

147 void f3**(**int a**[],** int b**[],** int ma**[][**n**],** int mb**[][**n**],** int mm**[][**n**],** int e**[])**

148 **{**

149 int bmamm**[**n**],** amb**[**n**];**

150 int mamm**[**n**][**n**];**

151 mul**(**ma**,** mm**,** mamm**);**

152 mul**(**b**,** mamm**,** bmamm**);**

153 sortVector**(**a**);**

154 mul**(**a**,** mb**,** amb**);**

155 add**(**bmamm**,** amb**,** e**);**

156 **}**

157

158 void inputVector**(**int v**[**n**])**

159 **{**

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

161 **{**

162 v**[**i**]** **=** 1**;**

163 **}**

164 **}**

165

166 void inputMatrix**(**int m**[**n**][**n**])**

167 **{**

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

169 **{**

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

171 **{**

172 m**[**i**][**j**]** **=** 1**;**

173 **}**

174 **}**

175 **}**

176

177 void outputVector**(**int v**[**n**])**

178 **{**

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

180 **{**

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

182 **{**

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

184 **}**

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

186 **}**

187 **}**

188

189 void outputMatrix**(**int m**[**n**][**n**])**

190 **{**

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

192 **{**

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

194 **{**

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

196 **{**

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

198 **}**

199 cout **<<** endl**;**

200 **}**

201 **}**

202 **}**

203

204 void sortVector**(**int v**[**n**])**

205 **{**

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

207 **}**

209 void add**(**int left**[],** int right**[],** int res**[])**

210 **{**

211 res**[**n**];**

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

213 **{**

214 res**[**i**]** **=** left**[**i**]** **+** right**[**i**];**

215 **}**

216 **}**

217

218 void transpose**(**int m**[][**n**],** int res**[][**n**])**

219 **{**

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

221 **{**

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

223 **{**

224 res**[**i**][**j**]** **=** m**[**j**][**i**];**

225 **}**

226 **}**

227 **}**

228

230 void mul**(**int left**[][**n**],** int right**[][**n**],** int res**[][**n**])**

231 **{**

232 res**[**n**][**n**];**

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

234 **{**

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

236 **{**

237 res**[**i**][**j**]** **=** 0**;**

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

239 **{**

240 res**[**i**][**j**]** **=** res**[**i**][**j**]** **+** left**[**i**][**k**]** **\*** right**[**k**][**j**];**

241 **}**

242 **}**

243 **}**

244 **}**

245

246 void mul**(**int left**[],** int right**[][**n**],** int res**[])**

247 **{**

248 res**[**n**];**

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

250 **{**

251 res**[**j**]** **=** 0**;**

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

253 **{**

254 res**[**j**]** **=** res**[**j**]** **+** left**[**k**]** **\*** right**[**k**][**j**];**

255 **}**

256 **}**

257 **}**