

```
1 // matrices_txt_stream.cpp
2 //
3 #ifdef _WIN32
4     #pragma warning(disable:4996)
5     #include <tchar.h>
6     #include <windows.h>
7     #include <conio.h>
8     #include <direct.h>
9 #elif (defined __linux__ || (defined _AIX))
10    #include <stdlib.h>
11    #include <sys/types.h>
12    #include <sys/stat.h>
13    #include <unistd.h>
14    typedef char _TCHAR;
15    #define _tmain main
16 #endif
17
18 #include <string.h>
19 #include <fstream>
20 #include <iostream>
21 #include <iomanip>
22 using namespace std;
23
24 #define MAXLINE 255
25 #define MAXLEN 20
26 typedef float MATTYPE[MAXLEN][MAXLEN];
27
28 void my_getch();
29 void GetNamesOfIOFiles(const char[], char[], const char[], char[], char[]);
30
31 int _tmain(int argc, _TCHAR* argv[])
32 {
33     ifstream in;
34     ofstream out;
35     FILE* fileC;
36     MATTYPE A, B, C;
37     float Suma;
38     int m, n, r, s, i, j;
39     char name[9], input_file[MAXLINE], output_file[MAXLINE],
40         pure_output_path[MAXLINE];
41
42     cout << "\nTento program nacita z textoveho suboru matice A a B\n";
43     cout << "a maticu C=A*B vypise na obrazovku a do textoveho suboru.\n\n";
44
45     GetNamesOfIOFiles("MATICE.TXT", input_file, "VYSLEDOK.TXT", output_file,
46         pure_output_path);
47     in.open(input_file, ios::in);
48     in >> m >> n;
49     if (in.fail()) {
50         cout << "Subor MATICE.TXT sa nepodarilo otvorit.\n";
51         my_getch();
52         return 1;
53     }
```

```
54     for (i = 0; i < m; i++)
55         for (j = 0; j < n; j++)
56             in >> A[i][j];
57     in >> r >> s;
58     if (n != r) {
59         cout << "Matice sa nedaju nasobit!";
60         my_getch();
61         return 1;
62     }
63     for (i = 0; i < r; i++)
64         for (j = 0; j < s; j++)
65             in >> B[i][j];
66     in.close();
67
68     for (i = 0; i < m; i++)
69         for (j = 0; j < s; j++) {
70             Suma = 0;
71             for (int k = 0; k < n; k++)
72                 Suma += A[i][k] * B[k][j];
73             C[i][j] = Suma;
74         }
75
76     cout << "Z textoveho suboru sa nacitali 2 matice.\n\nA:";
77     cout.setf(ios::fixed, ios::floatfield);
78     cout.precision(2);
79     for (i = 0; i < m; i++) {
80         for (j = 0; j < n; j++)
81             cout << setw(8) << A[i][j];
82         cout << "\n ";
83     }
84     cout << "\nB:";
85     for (i = 0; i < r; i++) {
86         for (j = 0; j < s; j++)
87             cout << setw(8) << B[i][j];
88         cout << "\n ";
89     }
90     cout << "\nVysledna matica C=A*B je:\n\n";
91     for (i = 0; i < m; i++) {
92         cout << " ";
93         for (j = 0; j < s; j++)
94             cout << setw(8) << C[i][j];
95         cout << "\n";
96     }
97
98     if ((fileC = fopen(output_file, "r")) != NULL) {
99         fclose(fileC);
100        cout << "\nVystupny subor VYSLEDOK.TXT uz existuje.\n\n";
101        cout << " 1. Chcete prepisat tento subor?\n";
102        cout << " 2. Chcete pripisat nove vysledky na koniec tohto suboru?\n";
103        cout << " 3. Chcete zadat nový názov pre vystupny subor?\n\n";
104        do {
105            cout << "Volba=";
106            cin >> i;
```

```
107     } while ((i < 1) || (i > 3));
108     switch (i) {
109     case 1: out.open(output_file, ios::out);
110             break;
111     case 2: out.open(output_file, ios::app);
112             break;
113     case 3: cout << "\nNapiste novy nazov suboru s maximalne 8 znakmi a ";
114             cout << "bez pripony: ";
115             scanf("%8s", name);
116             strcat(name, ".txt");
117             strcpy(output_file, pure_output_path);
118             strcat(output_file, name);
119             out.open(output_file, ios::out);
120     }
121 }
122 else
123     out.open(output_file, ios::out);
124
125 out << "Z textoveho suboru sa nacitali 2 matice.\n";
126 out.setf(ios::fixed, ios::floatfield);
127 out.precision(2);
128 out << "\nA:";
129 for (i = 0; i < m; i++) {
130     for (j = 0; j < n; j++)
131         out << setw(8) << A[i][j];
132     out << "\n ";
133 }
134 out << "\nB:";
135 for (i = 0; i < r; i++) {
136     for (j = 0; j < s; j++)
137         out << setw(8) << B[i][j];
138     out << "\n ";
139 }
140 out << "\nVysledna matica C=A*B je:\n";
141 out << "\nC:";
142 for (i = 0; i < m; i++) {
143     for (j = 0; j < s; j++)
144         out << setw(8) << C[i][j];
145     out << "\n ";
146 }
147 out << "\n";
148 out.close();
149
150 my_getch();
151 return 0;
152 }
153 //-----
154 void my_getch()
155 {
156 #ifdef _WIN32
157     _getch();
158 #else
159     cout << endl;
```

```
160 #endif
161 }
162 //-----
163 void GetNamesOfIOFiles(const char name_of_input_file[], char path_to_input_file[],
164     const char name_of_output_file[], char path_to_output_file[],
165     char pure_output_path[])
166 {
167     char current_path[MAXLINE];
168     current_path[0] = '\0';
169
170 #ifdef _WIN32
171     TCHAR exePath[MAXLINE];
172
173     HMODULE hModule = GetModuleHandle(NULL);
174     if (hModule != NULL) {
175         if (!GetModuleFileName(hModule, exePath, MAXLINE)) {
176             cout << "Nepodarila sa zistit cesta k exe-suboru.\n";
177             my_getch();
178             exit(1);
179         }
180     }
181     else {
182         cout << "Module handle is NULL.\n" << endl;
183         my_getch();
184         exit(1);
185     }
186
187     int iii;
188     bool flag = false;
189     for (iii = (int)wcslen(exePath); iii >= 0; iii--) {
190         if (!flag && exePath[iii] == '\\') {
191             current_path[iii + 1] = '\0';
192             flag = true;
193         }
194         if (flag)
195             current_path[iii] = (char)exePath[iii];
196     }
197 #elif (defined __linux__)
198     unsigned iii;
199     char line[MAXLINE];
200     FILE* fp;
201     if ((fp = popen("/bin/pwd", "r")) == NULL) {
202         perror("popen error");
203         exit(1);
204     }
205     if (fgets(line, MAXLINE, fp) == NULL) {
206         perror("fgets error");
207         exit(1);
208     }
209     pclose(fp);
210
211     iii = 0;
212     while (line[iii] != '\r' && line[iii] != '\n') {
```

```
213     current_path[iii] = line[iii];
214     iii++;
215 }
216 current_path[iii] = '\0';
217 #elif (defined _AIX)
218     unsigned iii;
219     char line[MAXLINE];
220     FILE* fp;
221     if ((fp = popen("user/bin/pwd", "r")) == NULL) {
222         perror("popen error");
223         exit(1);
224     }
225     if (fgets(line, MAXLINE, fp) == NULL) {
226         perror("fgets error");
227         exit(1);
228     }
229     pclose(fp);
230
231     iii = 0;
232     while (line[iii] != '\r' && line[iii] != '\n') {
233         current_path[iii] = line[iii];
234         iii++;
235     }
236     current_path[iii] = '\0';
237 #endif
238
239     path_to_input_file[0] = '\0';
240     strcat(path_to_input_file, current_path);
241     #if (defined __linux__) || (defined _AIX)
242         strcat(path_to_input_file, "/inputs/");
243     #elif (defined _WIN32)
244         strcat(path_to_input_file, "inputs\\");
245     #endif
246     strcat(path_to_input_file, name_of_input_file);
247
248     path_to_output_file[0] = '\0';
249     strcat(path_to_output_file, current_path);
250     #if (defined __linux__) || (defined _AIX)
251         struct stat st = { 0 };
252         strcat(path_to_output_file, "/outputs/");
253
254         if (stat(path_to_output_file, &st) == -1)
255             mkdir(path_to_output_file, 0755);
256     #elif (defined _WIN32)
257         strcat(path_to_output_file, "outputs\\");
258         if (_mkdir(path_to_output_file) != 0)
259             if (errno == ENOENT) {
260                 perror("_mkdir error");
261                 exit(1);
262             }
263     #endif
264     pure_output_path[0] = '\0';
265     strcpy(pure_output_path, path_to_output_file);
```

```
266     strcat(path_to_output_file, name_of_output_file);  
267 }  
268 //-----  
269
```