

```
1 // matrices_txt_ansi.cpp
2 //
3 #ifdef _WIN32
4     #pragma warning(disable:4996)
5     #include <tchar.h>
6     #include <conio.h>
7     #include <windows.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 <stdio.h>
19 #include <string.h>
20 #include <iostream>
21 using namespace std;
22
23 #define MAXLINE 255
24 #define MAXLEN 20
25 typedef float MATTYPERE[MAXLEN][MAXLEN];
26
27 void my_getch();
28 void GetNamesOfIOFiles(const char[], char[], const char[], char[], char[]);
29
30 int _tmain(int argc, _TCHAR* argv[])
31 {
32     FILE* in, * out;
33     MATTYPERE A, B, C;
34     float Suma;
35     int m, n, r, s, i, j;
36     char name[9], input_file[MAXLINE], output_file[MAXLINE],
37         pure_output_path[MAXLINE];
38
39     cout << "\nTento program nacita z textoveho suboru matice A a B\n";
40     cout << "a maticu C=A*B vypise na obrazovku a do textoveho suboru.\n\n";
41
42     GetNamesOfIOFiles("MATICE.TXT", input_file, "VYSLEDOK.TXT", output_file,
43         pure_output_path);
44     if ((in = fopen(input_file, "r")) == NULL) {
45         cout << "Subor MATICE.TXT sa nepodarilo otvorit.\n";
46         my_getch();
47         return 1;
48     }
49     fscanf(in, "%i%i", &m, &n);
50     for (i = 0; i < m; i++)
51         for (j = 0; j < n; j++)
52             fscanf(in, "%f", &A[i][j]);
53     fscanf(in, "%i%i", &r, &s);
```

```
54     if (n != r) {
55         cout << "Matice sa nedaju nasobit!";
56         my_getch();
57         return 1;
58     }
59     for (i = 0; i < r; i++)
60         for (j = 0; j < s; j++)
61             fscanf(in, "%f", &B[i][j]);
62     fclose(in);
63
64     for (i = 0; i < m; i++)
65         for (j = 0; j < s; j++) {
66             Suma = 0;
67             for (int k = 0; k < n; k++)
68                 Suma += A[i][k] * B[k][j];
69             C[i][j] = Suma;
70         }
71
72     cout << "Z textoveho suboru sa nacitali 2 matice.\n\nA:";
73     for (i = 0; i < m; i++) {
74         for (j = 0; j < n; j++)
75             printf("%8.2f", A[i][j]);
76         cout << "\n ";
77     }
78     cout << "\nB:";
79     for (i = 0; i < r; i++) {
80         for (j = 0; j < s; j++)
81             printf("%8.2f", B[i][j]);
82         cout << "\n ";
83     }
84     cout << "\nVysledna matica C=A*B je:\n\n";
85     for (i = 0; i < m; i++) {
86         cout << " ";
87         for (j = 0; j < s; j++)
88             printf("%8.2f", C[i][j]);
89         cout << "\n";
90     }
91
92     if ((out = fopen(output_file, "r")) != NULL) {
93         cout << "\nVystupny subor VYSLEDOK.TXT uz existuje.\n\n";
94         cout << " 1. Chcete prepisat tento subor?\n";
95         cout << " 2. Chcete pripisat nove vysledky na koniec tohto suboru?\n";
96         cout << " 3. Chcete zadat novy nazov pre vystupny subor?\n\n";
97         do {
98             cout << "Volba=";
99             cin >> i;
100         } while ((i < 1) || (i > 3));
101         switch (i) {
102         case 1: if ((out = fopen(output_file, "w")) == NULL) {
103             cout << "Subor VYSLEDOK.TXT sa nepodarilo otvorit.\n";
104             my_getch();
105             return 1;
106         }
```

```
107         else
108         break;
109     case 2: if ((out = fopen(output_file, "a")) == NULL) {
110         cout << "Subor VYSLEDOK.TXT sa nepodarilo otvorit.\n";
111         my_getch();
112         return 1;
113     }
114         else
115         break;
116     case 3: cout << "\nNapiste novy nazov suboru s maximalne 8 znakmi a ";
117         cout << "bez pripony: ";
118         scanf("%8s", name);
119         strcat(name, ".txt");
120         strcpy(output_file, pure_output_path);
121         strcat(output_file, name);
122         if ((out = fopen(output_file, "w")) == NULL) {
123             cout << "Subor " << name << " sa nepodarilo otvorit.\n";
124             my_getch();
125             return 1;
126         }
127     }
128 }
129 else
130     if ((out = fopen(output_file, "w")) == NULL) {
131         cout << "Subor VYSLEDOK.TXT sa nepodarilo otvorit.\n";
132         my_getch();
133         return 1;
134     }
135
136     fprintf(out, "Z textoveho suboru sa nacitali 2 matice.\n");
137     fprintf(out, "\nA:");
138     for (i = 0; i < m; i++) {
139         for (j = 0; j < n; j++)
140             fprintf(out, "%8.2f", A[i][j]);
141         fprintf(out, "\n ");
142     }
143     fprintf(out, "\nB:");
144     for (i = 0; i < r; i++) {
145         for (j = 0; j < s; j++)
146             fprintf(out, "%8.2f", B[i][j]);
147         fprintf(out, "\n ");
148     }
149     fprintf(out, "\nVysledna matica C=A*B je:\n\n");
150     for (i = 0; i < m; i++) {
151         fprintf(out, " ");
152         for (j = 0; j < s; j++)
153             fprintf(out, "%8.2f", C[i][j]);
154         fprintf(out, "\n");
155     }
156     fprintf(out, "\n");
157     fclose(out);
158
159     my_getch();
```

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

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

```
266     strcat(path_to_output_file, "outputs\\");
267     if (_mkdir(path_to_output_file) != 0)
268         if (errno == ENOENT) {
269             perror("_mkdir error");
270             exit(1);
271         }
272 #endif
273     pure_output_path[0] = '\\0';
274     strcpy(pure_output_path, path_to_output_file);
275     strcat(path_to_output_file, name_of_output_file);
276 }
277 //-----
278
```