```cpp
// dynamic_alocation_of_tensor_new.cpp
//
#ifdef _WIN32
   #pragma warning(disable:4996)
   #include <tchar.h>
   #include <windows.h>
   #include <conio.h>
   #include <direct.h>
#elif (defined __linux__) || (defined _AIX)
   #include <stdlib.h>
   #include <sys/types.h>
   #include <sys/stat.h>
   #include <unistd.h>
   typedef char _TCHAR;
   #define _tmain main
#endif

#include <stdio.h>
#include <string.h>
#include <iostream>
using namespace std;

#define MAXLINE 255
//#define MAXLEN 20    Nie je treba!

void my_getch();
void VypisTenzor(const unsigned, const unsigned, const unsigned, long double***);
void GetNamesOfIOFiles(const char[], char[]);
//---------------------------------------------------------------------------
int _tmain(int argc, _TCHAR* argv[])
{
   int i, j, k, m, n, o;
   long double*** T;
   FILE* in;
   char nameoffile[MAXLINE];

   cout << "\n  Program nacita z textoveho suboru tenzor T a vypise ho"
      << " na obrazovku.\n\n";

   GetNamesOfIOFiles("TENZOR.TXT", nameoffile);
   if ((in = fopen(nameoffile, "r")) == NULL) {
      cout << "  Subor TENZOR.TXT sa nepodarilo otvorit.\n";
      getch();
      return 1;
   }

   fscanf(in, "%i%i%i", &m, &n, &o);      // mnemotechnicka pomocka
   T = new long double** [m];          // pocet [] a * je 3
   for (i = 0; i < m; i++) {
      T[i] = new long double* [n];      // pocet [] a * je 3
      for (j = 0; j < n; j++)
         T[i][j] = new long double[o];  // pocet [] je 3
   }
```

```
54
55    for (i = 0; i < m; i++)
56       for (j = 0; j < n; j++)
57          for (k = 0; k < o; k++)
58             fscanf(in, "%Lf", &T[i][j][k]);
59
60    cout << "  Z textoveho suboru bol nacitany tenzor T radu m x n x o = " << m
61       << "x" << n << "x" << o << ":\n";
62    VypisTenzor(m, n, o, T);
63
64    for (i = 0; i < m; i++) { // Kazde new ma mat svoje delete
65       for (j = 0; j < n; j++)
66          delete[] T[i][j];
67       delete[] T[i];
68    }
69    delete[] T;
70
71    my_getch();
72    return 0;
73 }
74 //---------------------------------------------------------------------------
75 void my_getch()
76 {
77 #ifdef _WIN32
78    _getch();
79 #else
80    cout << endl;
81 #endif
82 }
83 //---------------------------------------------------------------------------
84 // Tenzor ako parameter funkcie volame odkazom "by dereference"
85 void VypisTenzor(const unsigned m, const unsigned n, const unsigned o,
86    long double*** X)
87 {
88    unsigned i, j, k;
89
90    for (i = 0; i < m; i++) {
91       cout << "\n  m = " << i + 1 << ":\n";
92       for (j = 0; j < n; j++) {
93          for (k = 0; k < o; k++)
94             printf("%12.6Lf", X[i][j][k]);
95          cout << "\n";
96       }
97    }
98 }
99 //---------------------------------------------------------------------------
100 void GetNamesOfIOFiles(const char name_of_input_file[], char path_to_input_file[])
101 {
102    char current_path[MAXLINE];
103    current_path[0] = '\0';
104
105 #ifdef _WIN32
106    TCHAR exePath[MAXLINE];
```

```
107
108     HMODULE hModule = GetModuleHandle(NULL);
109     if (hModule != NULL) {
110        if (!GetModuleFileName(hModule, exePath, MAXLINE)) {
111           cout << "Nepodarila sa zistit cesta k exe-suboru.\n";
112           my_getch();
113           exit(1);
114        }
115     }
116     else {
117        cout << "Module handle is NULL.\n" << endl;
118        my_getch();
119        exit(1);
120     }
121
122     int iii;
123     bool flag = false;
124     for (iii = (int)wcslen(exePath); iii >= 0; iii--) {
125        if (!flag && exePath[iii] == '\\') {
126           current_path[iii + 1] = '\0';
127           flag = true;
128        }
129        if (flag)
130           current_path[iii] = (char)exePath[iii];
131     }
132  #elif (defined __linux__)
133     unsigned iii;
134     char line[MAXLINE];
135     FILE* fp;
136     if ((fp = popen("/bin/pwd", "r")) == NULL) {
137        perror("popen error");
138        exit(1);
139     }
140     if (fgets(line, MAXLINE, fp) == NULL) {
141        perror("fgets error");
142        exit(1);
143     }
144     pclose(fp);
145
146     iii = 0;
147     while (line[iii] != '\r' && line[iii] != '\n') {
148        current_path[iii] = line[iii];
149        iii++;
150     }
151     current_path[iii] = '\0';
152  #elif (defined _AIX)
153     unsigned iii;
154     char line[MAXLINE];
155     FILE* fp;
156     if ((fp = popen("user/bin/pwd", "r")) == NULL) {
157        perror("popen error");
158        exit(1);
159     }
```

```
160     if (fgets(line, MAXLINE, fp) == NULL) {
161        perror("fgets error");
162        exit(1);
163     }
164     pclose(fp);
165
166     iii = 0;
167     while (line[iii] != '\r' && line[iii] != '\n') {
168        current_path[iii] = line[iii];
169        iii++;
170     }
171     current_path[iii] = '\0';
172  #endif
173
174     path_to_input_file[0] = '\0';
175     strcat(path_to_input_file, current_path);
176  #if (defined __linux__) || (defined _AIX)
177     strcat(path_to_input_file, "/inputs/");
178  #elif (defined _WIN32)
179     strcat(path_to_input_file, "inputs\\");
180  #endif
181     strcat(path_to_input_file, name_of_input_file);
182  }
183  //--------------------------------------------------------------------------
184
```