

```
1 #ifndef _WIN32
2     #pragma warning(disable:4996)
3     #include <conio.h>
4     #include <direct.h>
5 #elif (defined __linux__) || (defined _AIX)
6     #include <stdlib.h>
7     #include <sys/types.h>
8     #include <sys/stat.h>
9     #include <unistd.h>
10    typedef char _TCHAR;
11    #define _tmain main
12 #endif
13
14 #include <stdio.h>
15 #include <iostream>
16 #include <iomanip>
17 using namespace std;
18
19 #include "MaticaObdlznikova.h"
20 #include "MaticaStvorcova.h"
21 //-----
22 // This is a default constructor to initiate the base class 'TMaticaStvorcova'
23 // for childs
24 TMaticaStvorcova::TMaticaStvorcova()
25 {
26     Matica = NULL; // dolezite
27 }
28 //-----
29 //Konstruktor dynamicky alokuje pamat pre prvky matice a inicializuje ich
30 TMaticaStvorcova::TMaticaStvorcova(const unsigned n)
31 {
32     unsigned i,j;
33
34     PocetRiadkov = n;
35     PocetStlpcov = n;
36
37     Matica = new long double*[PocetRiadkov]; // pocet [] a * je 3
38     for (i=0;i<PocetRiadkov;i++)
39         Matica[i] = new long double[PocetStlpcov]; // pocet [] je 3
40
41     for (i=0; i<PocetRiadkov; i++)
42         for (j=0; j<PocetStlpcov; j++)
43             Matica[i][j]=0;
44 }
45 //-----
46 //Kopirovaci konstruktor
47 TMaticaStvorcova::TMaticaStvorcova(TMaticaStvorcova& X)
48 {
49     unsigned i,j;
50
51     PocetRiadkov = X.PocetRiadkov;
52     PocetStlpcov = X.PocetStlpcov;
53
```

```
54 Matica = new long double*[PocetRiadkov]; // pocet [] a * je 3
55 for (i=0;i<PocetRiadkov;i++)
56     Matica[i] = new long double[PocetStlpcov]; // pocet [] je 3
57
58 for (i=0;i<PocetRiadkov;i++)
59     for (j=0;j<PocetStlpcov;j++)
60         Matica[i][j] = X.Matica[i][j];
61 }
62 //-----
63 //Destruktor upratuje alokovanu pamat
64 TMaticaStvorcova::~TMaticaStvorcova()
65 {
66     if (Matica != NULL) {
67         for (unsigned i=0;i<PocetRiadkov;i++)
68             delete[] Matica[i];
69         delete[] Matica;
70         Matica = NULL;
71     }
72 }
73 //-----
74 TMaticaStvorcova& TMaticaStvorcova::operator=(const TMaticaStvorcova& X)
75 {
76     if (this == &X) return *this;
77
78     for (unsigned i=0;i<PocetRiadkov;i++)
79         for (unsigned j=0;j<PocetStlpcov;j++)
80             Matica[i][j] = X.Matica[i][j];
81
82     return *this;
83 }
84 //-----
85 // Ak mame C = A + B a A a B su stvorcove matice, potom A + B je obdlznikova matica,
86 // zatiaľ čo C je stvorcova matica. Preto potrebujeme nasledujuce pretazenie:
87 TMaticaStvorcova& TMaticaStvorcova::operator=(const TMaticaObdlznikova& X)
88 {
89     if (this == &X) return *this;
90
91     my_class xx;
92
93     if (X.getPocetRiadkov() != X.getPocetStlpcov()){
94         cout << "\nTato obdlznikova matica sa neda previest na stvorcovu!";
95         xx.my_getch();
96         exit(1);
97     }
98
99     for (unsigned i=0;i<PocetRiadkov;i++)
100         for (unsigned j=0;j<PocetStlpcov;j++)
101             Matica[i][j] = X.getPrvokMatice(i,j);
102
103     return *this;
104 }
105 //-----
106 TMaticaStvorcova TMaticaStvorcova::UmocniMaticu(unsigned k)
```

```
107 {
108     TMaticaStvorcova Y(PocetRiadkov), PomMatica0(PocetRiadkov),
109         PomMatica(PocetRiadkov);
110
111     switch (k) {
112     case 0:
113         for (unsigned i = 0; i<PocetRiadkov; i++) {
114             for (unsigned j = 0; j<PocetRiadkov; j++)
115                 Y.Matica[i][j] = 0;
116             Y.Matica[i][i] = 1;
117         }
118         break;
119     case 1:
120         Y = *this;
121         break;
122     default:
123         PomMatica0 = PomMatica = *this;
124         k--;
125         while (k-->0) {
126             Y = PomMatica * PomMatica0;
127             PomMatica = Y;
128         }
129     }
130
131     return Y;
132 }
133 //-----
134
135
```