```cpp
#ifdef _WIN32
    #pragma warning(disable:4996)
    #include <conio.h>
    #include <direct.h>
#elif (defined __linux__) || (defined _AIX) || (defined __APPLE__)
    #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 <iostream>
#include <iomanip>
using namespace std;

#include "MaticaObdlznikova.h"
#include "MaticaStvorcova.h"
//---------------------------------------------------------------------------
// This is a default constructor to initiate the base class 'TMaticaStvorcova'
// for childs
TMaticaStvorcova::TMaticaStvorcova()
{
    Matica = NULL; // dolezite
}
//---------------------------------------------------------------------------
//Konstruktor dynamicky alokuje pamat pre prvky matice a inicializuje ich
TMaticaStvorcova::TMaticaStvorcova(const unsigned n)
{
    unsigned i,j;

    PocetRiadkov = n;
    PocetStlpcov = n;

    Matica = new long double*[PocetRiadkov];         // pocet [] a * je 3
    for (i=0;i<PocetRiadkov;i++)
        Matica[i] = new long double[PocetStlpcov];    // pocet [] je 3

    for (i=0; i<PocetRiadkov; i++)
        for (j=0; j<PocetStlpcov; j++)
            Matica[i][j]=0;
}
//---------------------------------------------------------------------------
//Kopirovaci konstruktor
TMaticaStvorcova::TMaticaStvorcova(TMaticaStvorcova& X)
{
    unsigned i,j;

    PocetRiadkov = X.PocetRiadkov;
    PocetStlpcov = X.PocetStlpcov;

    Matica = new long double*[PocetRiadkov];         // pocet [] a * je 3
    for (i=0;i<PocetRiadkov;i++)
        Matica[i] = new long double[PocetStlpcov];    // pocet [] je 3

    for (i=0;i<PocetRiadkov;i++)
        for (j=0;j<PocetStlpcov;j++)
            Matica[i][j] = X.Matica[i][j];
}
//---------------------------------------------------------------------------
//Destruktor upratuje alokovanu pamat
TMaticaStvorcova::~TMaticaStvorcova()
{
    if (Matica != NULL) {
        for (unsigned i=0;i<PocetRiadkov;i++)
            delete[] Matica[i];
        delete[] Matica;
        Matica = NULL;
    }
}
//---------------------------------------------------------------------------
```

```
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  // zatial co 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--) {
126             Y = PomMatica * PomMatica0;
127             PomMatica = Y;
128         }
129     }
130
131     return Y;
132 }
133 //---------------------------------------------------------------------------
134
135
```