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

    PocetRiadkov = m;
    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
TMaticaObdlznikova::TMaticaObdlznikova(TMaticaObdlznikova& 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
TMaticaObdlznikova::~TMaticaObdlznikova()
{
    if (Matica != NULL) {
        for (unsigned i=0;i<PocetRiadkov;i++)
            delete[] Matica[i];
        delete[] Matica;
        Matica = NULL;
    }
}
//---------------------------------------------------------------------------
TMaticaObdlznikova& TMaticaObdlznikova::operator=(const TMaticaObdlznikova& X)
```

```
74  {
75      if (this == &X) return *this;
76
77      for (unsigned i=0;i<PocetRiadkov;i++)
78          for (unsigned j=0;j<PocetStlpcov;j++)
79              Matica[i][j] = X.Matica[i][j];
80
81      return *this;
82  }
83  //---------------------------------------------------------------------------
84  const TMaticaObdlznikova operator+(const TMaticaObdlznikova& LavaMatica,
85      const TMaticaObdlznikova& PravaMatica)
86  {
87      TMaticaObdlznikova VyslMatica(LavaMatica.PocetRiadkov, LavaMatica.PocetStlpcov);
88
89      for (unsigned i=0; i<LavaMatica.PocetRiadkov; i++)
90          for (unsigned j=0; j<LavaMatica.PocetStlpcov; j++)
91              VyslMatica.Matica[i][j] =
                LavaMatica.Matica[i][j]+PravaMatica.Matica[i][j];
92
93      return VyslMatica;
94  }
95  //---------------------------------------------------------------------------
96  const TMaticaObdlznikova operator*(const TMaticaObdlznikova& LavaMatica,
97      const TMaticaObdlznikova& PravaMatica)
98  {
99      TMaticaObdlznikova VyslMatica(LavaMatica.PocetRiadkov, LavaMatica.PocetStlpcov);
100     my_class xx;
101
102     unsigned i,j,k;
103    long double Suma;
104
105    if (LavaMatica.PocetStlpcov!=PravaMatica.PocetRiadkov){
106         cout << "\n  Matice sa nedaju nasobit!\n";
107         xx.my_getch();
108     }
109    for (i=0; i<LavaMatica.PocetRiadkov; i++)
110         for (j=0; j<PravaMatica.PocetStlpcov; j++){
111             Suma=0;
112             for (k=0; k<LavaMatica.PocetStlpcov; k++)
113                 Suma+=LavaMatica.Matica[i][k]*PravaMatica.Matica[k][j];
114             VyslMatica.Matica[i][j]=Suma;
115         }
116     return VyslMatica;
117 }
118 //---------------------------------------------------------------------------
119 TMaticaObdlznikova& TMaticaObdlznikova::operator+=(const TMaticaObdlznikova&
        PravaMatica)
120 {
121     *this = *this + PravaMatica;
122     return *this;
123 }
124 //---------------------------------------------------------------------------
125 TMaticaObdlznikova& TMaticaObdlznikova::operator*=(const TMaticaObdlznikova&
        PravaMatica)
126 {
127     *this = *this * PravaMatica;
128     return *this;
129 }
130 //---------------------------------------------------------------------------
131 istream& operator>>(istream& is, TMaticaObdlznikova& X)
132 {
133     my_class xx;
134
135     for (unsigned i=0;i<X.PocetRiadkov;i++)
136         for (unsigned j=0;j<X.PocetStlpcov;j++)
137             is >> X.Matica[i][j];
138     if (is.fail()){
139         cout << "\n\n  Vstupny subor sa nepodarilo otvorit.\n";
140         xx.my_getch();
141         exit(1);
142     }
143
144
145
```

```
146         return is;
147     }
148     //---------------------------------------------------------------------------
149     ostream& operator<<(ostream& os,const TMaticaObdlznikova& X)
150     {
151         os.setf(ios::fixed,ios::floatfield);
152         os.precision(2);
153         for (unsigned i=0;i<X.PocetRiadkov;i++) {
154             os << "\n  ";
155             for (unsigned j=0;j<X.PocetStlpcov;j++)
156                 os << setw(8) << X.Matica[i][j];
157         }
158         os << "\n";
159 
160         return os;
161     }
162     //---------------------------------------------------------------------------
163     void my_class::my_getch() const
164     {
165     #ifdef _WIN32
166         _getch();
167     #else
168         cout << endl;
169     #endif
170     }
171     //---------------------------------------------------------------------------
172 
173 
```