```cpp
// complex_numbers.cpp
//
#ifdef _WIN32
    #include <tchar.h>
    #include <conio.h>
#elif (defined __linux__) || (defined _AIX) || (defined __APPLE__)
    typedef char _TCHAR;
    #define _tmain main
#endif

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

void my_getch();
//---------------------------------------------------------------------------
// Uzivatelsky definovana trieda
class TCisloKomplexne {
    float Re, Im; // defaultna vizibilita triedy class je private
public:
    TCisloKomplexne();
    void setCisloKomplexne(const float&, const float&); // const;
    TCisloKomplexne SucetKomplCisel(const TCisloKomplexne);
    // nasledujuca deklaracia sa berie ako CLENSKA funkcia
    //  const TCisloKomplexne operator+(const TCisloKomplexne&) const;
    // nasledujuca deklaracia sa berie ako NECLENSKA funkcia
    friend const TCisloKomplexne operator+(const TCisloKomplexne&,
        const TCisloKomplexne&);
    // pretazenie vstupneho alebo vystupneho prudu je vzdy NECLENSKA funkcia
    friend ostream& operator<<(ostream&, const TCisloKomplexne&);
    // friend spristupni private a protected polozky triedy neclenskej funkcii
};
//---------------------------------------------------------------------------
TCisloKomplexne::TCisloKomplexne()
{
    Re = 0;
    Im = 0;
}
//---------------------------------------------------------------------------
void TCisloKomplexne::setCisloKomplexne(const float& _re, const float& _im) // const
{
    Re = _re;
    Im = _im;
}
//---------------------------------------------------------------------------
TCisloKomplexne TCisloKomplexne::SucetKomplCisel(const TCisloKomplexne right)
{
    TCisloKomplexne z;

    z.Re = Re + right.Re;
    z.Im = Im + right.Im;

    return z;
}
//---------------------------------------------------------------------------
/* // Pretazenie operatora scitania + ako clenskej funkcie -- NEODPORUCA SA --
const TCisloKomplexne TCisloKomplexne::operator+(const TCisloKomplexne& right) const
{
TCisloKomplexne z;

z.Re = Re + right.Re;
z.Im = Im + right.Im;

// Nasledujuce 2 riadky robia to same ako predchadzajuce 2 riadky
//  z.Re = (*this).Re + right.Re;
//  z.Im = this->Im + right.Im;

return z;
} */
//---------------------------------------------------------------------------
// Pretazenie operatora scitania + podla syntaxu v tabulke
// Telo pretazenia je identicke s telom predchadzajucej funkcie, iba hlavicka je ina
```

```
74  const TCisloKomplexne operator+(const TCisloKomplexne& left,
75      const TCisloKomplexne& right)
76  {
77      TCisloKomplexne z;
78
79      z.Re = left.Re + right.Re;
80      z.Im = left.Im + right.Im;
81
82      return z;
83  }
84  //-----------------------------------------------------------------------------
85  // Pretazenie operatora vystupneho prudu <<
86  ostream& operator<<(ostream& os, const TCisloKomplexne& com)
87  {
88      os.setf(ios::fixed, ios::floatfield);  // nastavenie formatu
89      os.precision(4);                       // nastavenie poctu desatinnych miest
90      os << setw(6) << com.Re;               // nastavenie sirky vypisu
91      if (com.Im != 0) {
92          os.setf(ios::showpos);               // setf, precision, unsetf su operatory
93          os << setw(6) << com.Im << 'i';      // setw je manipulator
94          os.unsetf(ios::showpos);             // setw potrebuje #include <iomanip.h>
95      }
96      return os;
97  }
98  //-----------------------------------------------------------------------------
99  void my_getch()
100 {
101 #ifdef _WIN32
102     _getch();
103 #else
104     cout << endl;
105 #endif
106 }
107 //-----------------------------------------------------------------------------
108 int _tmain(int argc, _TCHAR* argv[])
109 {
110     // TCisloKomplexne je trieda, t.j., typ premennej
111     // kompl_cislo[0], kompl_cislo[1], kompl_cislo[2],vysledok1 a vysledok2 su objekty
112     // alebo instancie triedy TCisloKomplexne, t.j. premenne typu TCisloKomplexne
113     unsigned i;
114     float re, im;
115     TCisloKomplexne kompl_cislo[3], vysledok1, vysledok2;
116
117     cout << "\nKonstruktor nastavil pociatocne hodnoty vsetkych komplexnych cisel:";
118     for (i = 0; i < 3; i++)
119         cout << "\nkompl_cislo[" << i << "] = " << kompl_cislo[i];
120     cout << "\nvysledok1 = " << vysledok1;
121     cout << "\nvysledok2 = " << vysledok2;
122
123     cout << "\n\nZ klavesnice nacitajte 2 komplexne cisla a na obrazovku"
124         << " vypiste ich sucet.\n\n";
125
126     cout << "Zadajte dve komplexne cisla ich realnymi (Re) a imaginarnymi"
127         << " (Im)zlozkami:\n";
128     for (i = 0; i < 2; i++) {
129         cout << "kompl_cislo[" << i << "]: Re = ";
130         cin >> re;
131         cout << "kompl_cislo[" << i << "]: Im = ";
132         cin >> im;
133         kompl_cislo[i].setCisloKomplexne(re, im);
134         cout << endl;
135     }
136
137     vysledok1 = kompl_cislo[0].SucetKomplCisel(kompl_cislo[1]);
138     cout << "vysledok1 = kompl_cislo[0].SucetKomplCisel(kompl_cislo[1]); dava:\n";
139     cout << kompl_cislo[0] << " + " << kompl_cislo[1] << " = " << vysledok1;
140
141     vysledok2 = kompl_cislo[0] + kompl_cislo[1];
142     cout << "\n\nvysledok2 = kompl_cislo[0] + kompl_cislo[1]; dava:\n";
143     cout << kompl_cislo[0] << " + " << kompl_cislo[1] << " = " << vysledok2;
144
145     cout << "\n\nMedzi objekty mozeme davat prikaz priradenia =. Medzi polami to
        nejde!!!\n\n";
```

```
146         kompl_cislo[2] = kompl_cislo[0];
147         cout << "Prikaz kompl_cislo[2] = kompl_cislo[0] dava:\n";
148         cout << "kompl_cislo[2] = " << kompl_cislo[2] << endl;
149 
150         my_getch();
151         return 0;
152     }
153     //---------------------------------------------------------------------------
```