

```
1 // complex_numbers.cpp
2 //
3 #ifdef _WIN32
4     #include <tchar.h>
5     #include <conio.h>
6 #elif (defined __linux__) || (defined _AIX)
7     typedef char _TCHAR;
8     #define _tmain main
9 #endif
10
11 #include <stdio.h>
12 #include <iostream>
13 #include <iomanip>
14 using namespace std;
15
16 void my_getch();
17 //-----
18 // Uzivatelsky definovana trieda
19 class TCisloKomplexne {
20     float Re, Im; // defaultna vizibilita triedy class je private
21 public:
22     TCisloKomplexne();
23     void setCisloKomplexne(const float&, const float&); // const;
24     TCisloKomplexne SucetKomplCisel(const TCisloKomplexne);
25     // nasledujuca deklaracia sa berie ako CLENSKA funkcia
26     // const TCisloKomplexne operator+(const TCisloKomplexne&) const;
27     // nasledujuca deklaracia sa berie ako NECLENSKA funkcia
28     friend const TCisloKomplexne operator+(const TCisloKomplexne&,
29         const TCisloKomplexne&);
30     // pretazenie vstupneho alebo vystupneho prudu je vzdy NECLENSKA funkcia
31     friend ostream& operator<<(ostream&, const TCisloKomplexne&);
32     // friend sprístupni private a protected položky triedy neclenskej funkcii
33 };
34 //-----
35 TCisloKomplexne::TCisloKomplexne()
36 {
37     Re = 0;
38     Im = 0;
39 }
40 //-----
41 void TCisloKomplexne::setCisloKomplexne(const float& _re, const float& _im) // const
42 {
43     Re = _re;
44     Im = _im;
45 }
46 //-----
47 TCisloKomplexne TCisloKomplexne::SucetKomplCisel(const TCisloKomplexne right)
48 {
49     TCisloKomplexne z;
50
51     z.Re = Re + right.Re;
52     z.Im = Im + right.Im;
53
```

```
54     return z;
55 }
56 //-----
57 /* // Pretazenie operatora scitania + ako clenскеj funkcie -- NEODPORUCA SA --
58 const TCisloKomplexne TCisloKomplexne::operator+(const TCisloKomplexne& right) const
59 {
60     TCisloKomplexne z;
61
62     z.Re = Re + right.Re;
63     z.Im = Im + right.Im;
64
65     // Nasledujuce 2 riadky robia to same ako predchadzajuce 2 riadky
66     // z.Re = (*this).Re + right.Re;
67     // z.Im = this->Im + right.Im;
68
69     return z;
70 } */
71 //-----
72 // Pretazenie operatora scitania + podla syntaxu v tabulke
73 // 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 instance 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 nacistajte 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
146         nejde!!!\n\n";
147     kompl_cislo[2] = kompl_cislo[0];
148     cout << "Prikaz kompl_cislo[2] = kompl_cislo[0] dava:\n";
149     cout << "kompl_cislo[2] = " << kompl_cislo[2] << endl;
150
151     my_getch();
152     return 0;
153 }
154 //-----
```