

```

1 // complex_numbers.cpp
2 //
3 // Pretezovani operatoru pro tridu cplx - komplexni cisla
4 // Chcete-li tento soubor prekladat na nekterem ze starsich prekladacu, odstrante
5 // komentar u nasledujiciho prikazu:
6 // typedef int bool;
7
8 // UNARNI OPERATORY
9 // Operator minus (unarni): vrati kopii instance s obracenymi znamenkou slozek
10 // Je zde dvakrat, jednou jako metoda (uzavrena do komentare), podruhe jako
11 // spratelenou funkce - obe funguji, v programu smi byt ale jen jedna
12 // Operator + (unarni): vrati aktualni instanci beze zmeny
13 // Operator !: vrati true, je-li re == 0 a zaroven im == 0, jinak vrati false
14
15 // BINARNI OPERATORY
16 // Operator + (scitani)
17 // Operator * nasobeni
18 #ifdef _WIN32
19     #include <tchar.h>
20     #include <conio.h>
21 #elif (defined __linux__) || (defined _AIX) || (defined __APPLE__)
22     typedef char _TCHAR;
23     #define _tmain main
24 #endif
25
26 #include <math.h>
27 #include <iostream>
28 using namespace std;
29
30 class cplx {
31 private:
32     double re, im;
33 public:
34     cplx(double r = 0, double i = 0) : re(r), im(i) {}
35     double Re() { return re; }
36     double Im() { return im; }
37     friend double Abs(cplx& c) { return sqrt(c.re * c.re + c.im * c.im); }
38
39     // UNARNI OPERATORY
40     // Operator unarni minus (otoci znamenska obou slozek)
41     // jako metoda bez parametru
42     //cplx operator-() { return cplx(-re, -im); } // = THE BEST WAY !!!!!!!!!!!
43
44     // Jina moznost: spratelenou funkce s jednim parametrem
45     friend cplx operator-(cplx& c);
46
47     bool operator!() { return !re && !im; } // = THE BEST WAY !!!!!!!!!!!
48     cplx operator+() { return *this; } // = THE BEST WAY !!!!!!!!!!!
49
50     // BINARNI OPERATORY
51     // Scitani jako metoda: funguje, ale ma sve problemy, pouzijeme radeji
52     // spratelenou funkci
53     // cplx operator+(cplx b) { return cplx(re+b.re, im+b.im);}
54     // nezvladne c = 5 + a; !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
55     friend cplx operator+(cplx a, cplx b); // = THE BEST WAY !!!!!!!!!!!
56     friend cplx operator*(cplx a, cplx b); // = THE BEST WAY !!!!!!!!!!!
57 };
58
59 // Jina moznost, jak deklarovat unarni minus pro komplexni cisla
60 cplx operator-(cplx& c)
61 {
62     return cplx(-c.re, -c.im);
63 }
64
65 // Jina moznost, jak deklarovat scitani pro komplexni cisla
66 cplx operator*(cplx a, cplx b)
67 {
68     return cplx(a.re * b.re - a.im * b.im, a.re * b.im + a.im * b.re);
69 }
70
71 /*
72 cplx cplx::operator+(cplx b)
73 {

```

```

74     return cplx(re+b.re, im+b.im);
75 } */
76 /*
77 cplx cplx::operator+(cplx a, cplx b)
78 {
79     return cplx(a.re+b.re, a.im+b.im);
80 } */
81
82 cplx operator+(cplx a, cplx b)
83 {
84     return cplx(a.re + b.re, a.im + b.im);
85 }
86
87 /*
88 // Znovu operator scitani, pochybna implementace
89 //NEFUNGUJE VE SLOZITEJSICH VYRAZECH - NENI REENTRANTNI
90 cplx& operator+(cplx &a, cplx &b)
91 {
92     static cplx pom;
93     pom = cplx(a.re+b.re, a.im+b.im);
94     return pom;
95 }
96 */
97
98 int my_getch();
99
100 int main()
101 {
102     cplx a(1, 1), b, i(0, 1), j(1, 0);
103     b = -a; // Vola se cplx::operator -()
104     bool c = !a;
105     c = !cplx(0, 1);
106     b = a + i;
107     b = a + 5;
108     b = 5 + a;
109     b = (a + j) * (i + j);
110
111     cout << "\n There is no output in this example.\n";
112     my_getch();
113     return 0;
114 }
115 //-----
116 int my_getch()
117 {
118     #ifdef _WIN32
119         _getch();
120     #else
121         cout << endl;
122     #endif
123     return 0;
124 }
125 //-----
126

```