

```
1 // field1.cpp
2 //
3 // Priklad pretizeneho operatoru indexovani []
4 // Jde o rozsireni prikladu ze souboru POLE2.CPP
5
6 // Zmeny:
7 // - nepojmenovany vyctovy typ je verejny, abychom mohli pouzivat
8 //   konstantu delka i mimo tridu
9 // - metodu Prvek() nahradil pretizeny operator []
10
11
12 // Vzhledem k rozdilum v chovani prekladacu
13 // zde nekontrolujeme, zda se alokace podarila, stejne jako
14 // v predchozich verzich
15
16 #ifdef _WIN32
17     #include <tchar.h>
18     #include <conio.h>
19 #elif (defined __linux__) || (defined _AIX)
20     typedef char _TCHAR;
21     #define _tmain main
22 #endif
23
24 #include <iostream>
25 using namespace std;
26
27 class pole {
28     int* p;
29 public:
30     enum { delka = 10 };
31     explicit pole(int d = 0);
32     pole(pole&);
33     int& operator[](int);
34     ~pole() { delete p; }
35 };
36
37 pole::pole(int d)
38 {
39     p = new int[delka];
40     for (int i = 0; i < delka; i++)
41         p[i] = d;
42 }
43
44 pole::pole(pole& P)
45 {
46     p = new int[delka];
47     for (int i = 0; i < delka; i++)
48         p[i] = P.p[i];
49 }
50
51 int& pole::operator[](int index)
52 {
53     return p[index];
```

```
54 }
55
56 void my_getch();
57
58 int main()
59 {
60     pole P(0);
61     cout << endl << "Pole P po konstruktore:\n";
62     for (int i = 0; i < pole::delka; i++) // Nyni muzeme pouziti pole::delka,
63         cout << P[i] << endl;         // nebo tento typ je verejny
64
65     for (int i = 0; i < pole::delka; i++)
66         P[i] = 10;
67     cout << "\nPole P po priradovani:\n";
68     for (int i = 0; i < pole::delka; i++)
69         cout << P[i] << endl;
70
71     pole Q(P);
72     cout << "\nPole Q po pouziti kopirovaciho konstruktora:\n";
73     for (int i = 0; i < pole::delka; i++)
74         cout << Q[i] << endl;
75     my_getch();
76     return 0;
77 }
78 //-----
79 void my_getch()
80 {
81     #ifdef _WIN32
82         _getch();
83     #else
84         cout << endl;
85     #endif
86 }
87 //-----
88
```