```cpp
// field1.cpp
//
// Priklad pretizeneho operatoru indexovani []
// Jde o rozsireni prikladu ze souboru POLE2.CPP

// Zmeny:
// - nepojmenovany vyctovy typ je verejny, abychom mohli pouzivat
//   konstantu delka i mimo tridu
// - metodu Prvek() nahradil pretizeny operator []


// Vzhledem k rozdilum v chovani prekladacu
// zde nekontrolujeme, zda se alokace podarila, stejne jako
// v predchozich verzich

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

#include <iostream>
using namespace std;

class pole {
    int* p;
public:
    enum { delka = 10 };
    explicit pole(int d = 0);
    pole(pole&);
    int& operator[](int);
    ~pole() { delete p; }
};

pole::pole(int d)
{
    p = new int[delka];
    for (int i = 0; i < delka; i++)
        p[i] = d;
}

pole::pole(pole& P)
{
    p = new int[delka];
    for (int i = 0; i < delka; i++)
        p[i] = P.p[i];
}

int& pole::operator[](int index)
{
    return p[index];
}

void my_getch();

int main()
{
    pole P(0);
    cout << endl << "Pole P po konstruktore:\n";
    for (int i = 0; i < pole::delka; i++) // Nyni muzeme pouzit pole::delka,
        cout << P[i] << endl;                 // nebot tento typ je verejny

    for (int i = 0; i < pole::delka; i++)
        P[i] = 10;
    cout << "\nPole P po priradovani:\n";
    for (int i = 0; i < pole::delka; i++)
        cout << P[i] << endl;

    pole Q(P);
    cout << "\nPole Q po pouziti kopirovaciho konstruktora:\n";
    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
```