II. Základné elementy

7. Dátové štruktúry / Polia (Arrays)

V programovacích jazykoch:
Postupnosť viacerých elementov (alebo objektov) rovnakého typu.

Príkazy

- Parametre príkazového riadku: **`args`**
  `args[0], args[1], args[2], ...`
- Lineárna algebra
  - Vektor **`vektor`**
    `vektor[0], vektor[1], vektor[2]`
  - Matica **`matrix`**
    `matrix[i][j]`

Príklady (pokrač.)

- Všeobecne: Indexované premenné:

x_i **x[i]**

```
double sum = 0.0;
for ( int i = 1; i <= n; i++)
{
  sum = sum + x[i];
}
```

$$\sum_{i=1}^{n} x_i$$

7.1 Polia: Deklarácia v Jave

Príklady:

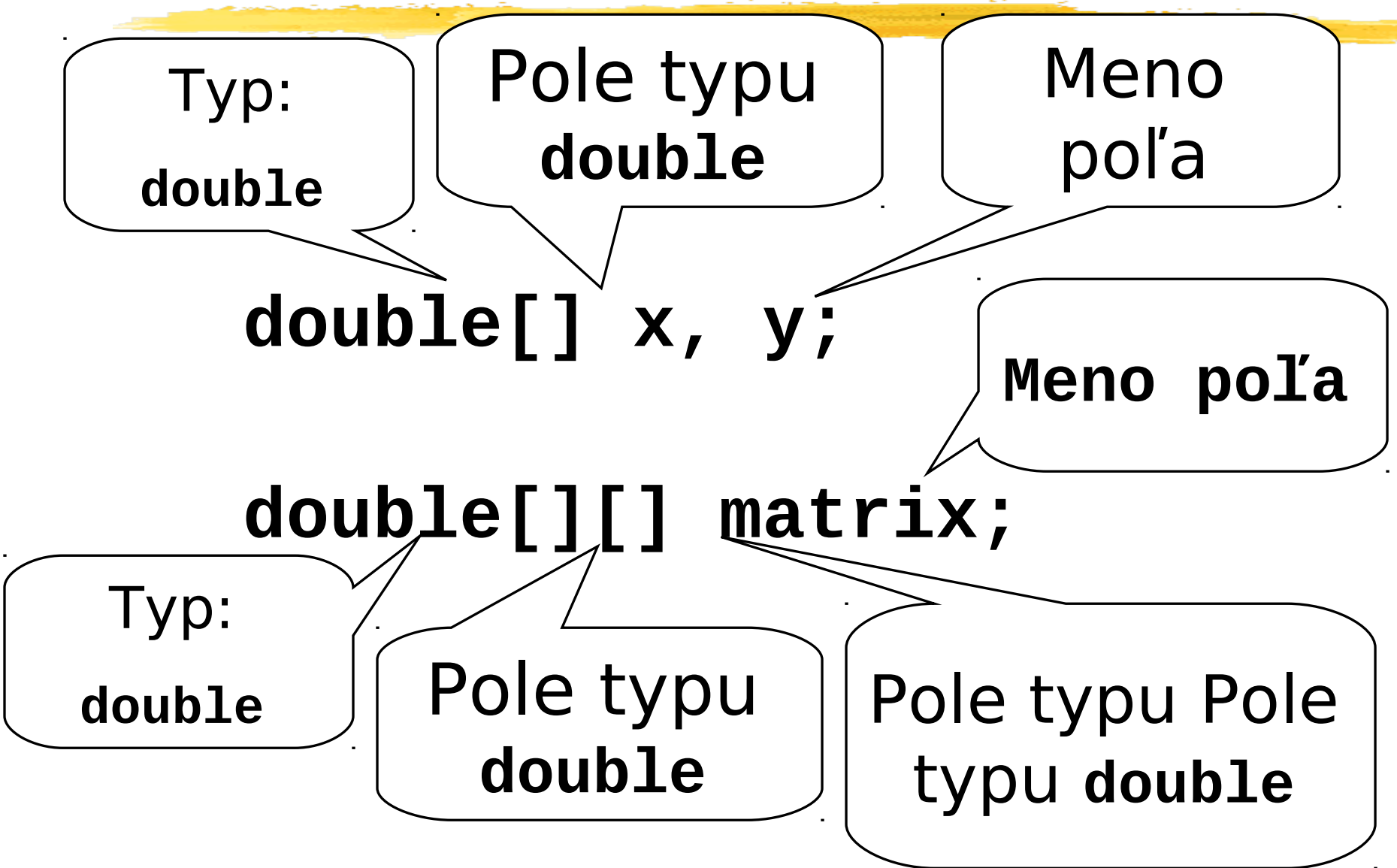
```
String[] args;   // Pole znakových reťazcov
String args[];   // to isté
double[] x, y;   // dve polia typu
     // double (Vektory)
double[][] matrix;  // Jedno dvojdimenzionálne
     // pole typu double (Matica)
int[][] k; // Dvojdimenzionálne pole typu
     // int
double[] vektor; // ďalší Vektor
```

7.1 Polia: Deklarácia v Jave

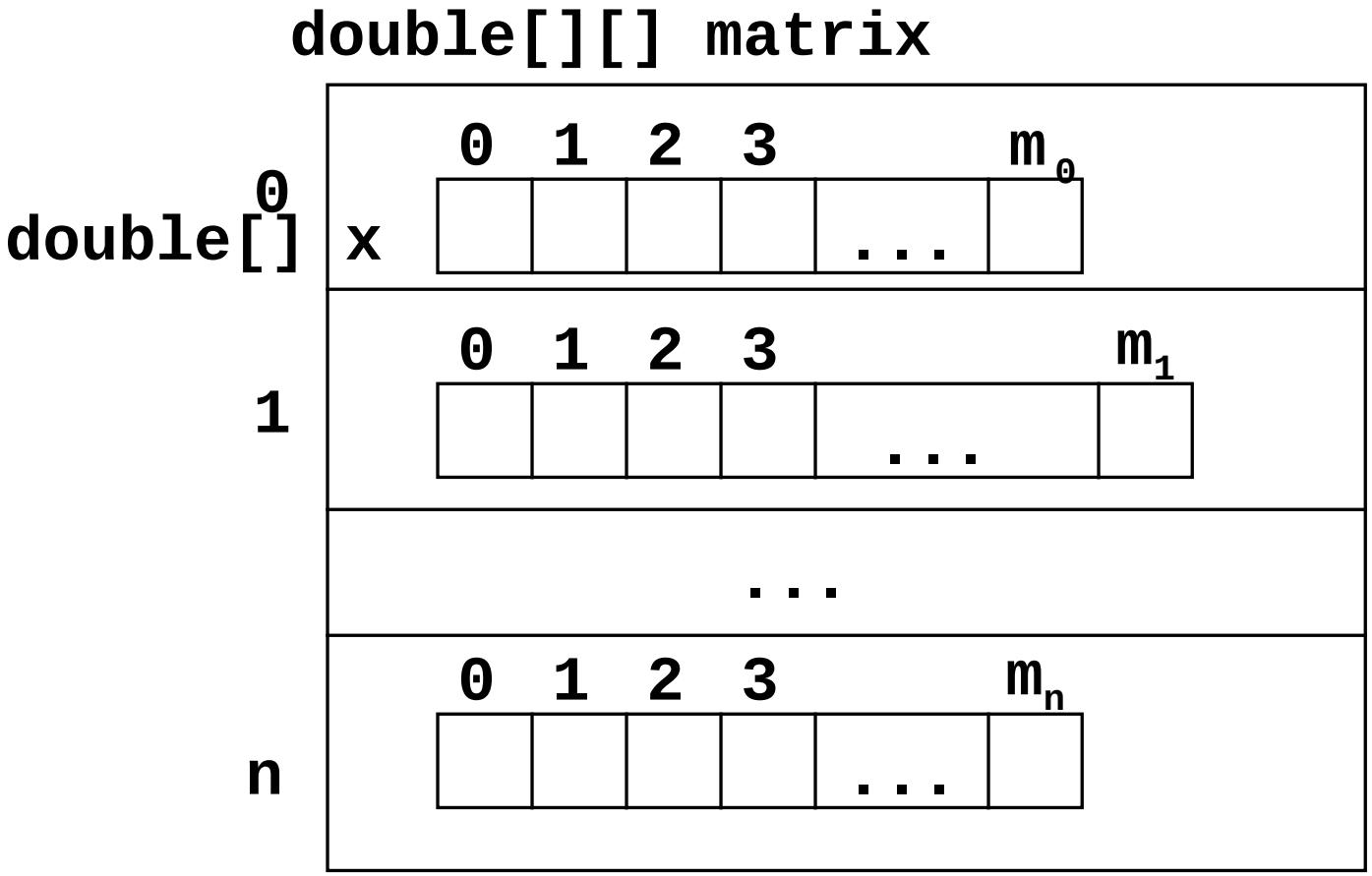
Príklady:

```
String[] args;   // Pole znakových reťazcov
String args[];   // to isté
double[] x, y;   // dve polia typu
     // double (Vektory)
double[][] matrix;  // Jedno dvojdimenzionálne
     // pole typu double (Matica)
int[][] k; // Dvojdimenzionálne pole typu
     // int
double[] vektor; // ďalší Vektor
```

Princíp (Java)



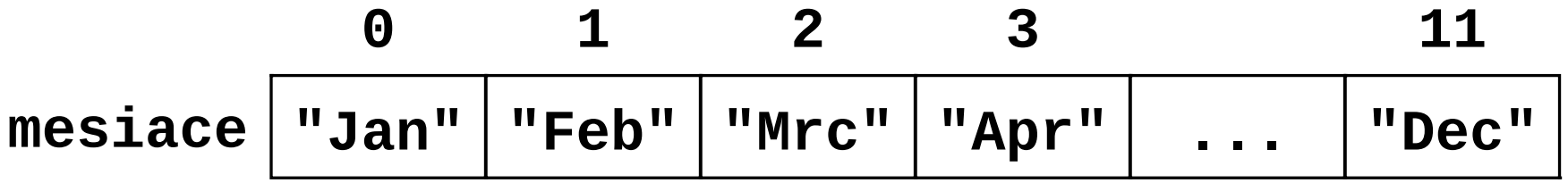
Viacdimenzionálne polia (Java)



7.2 Polia: Priradenie (Java)

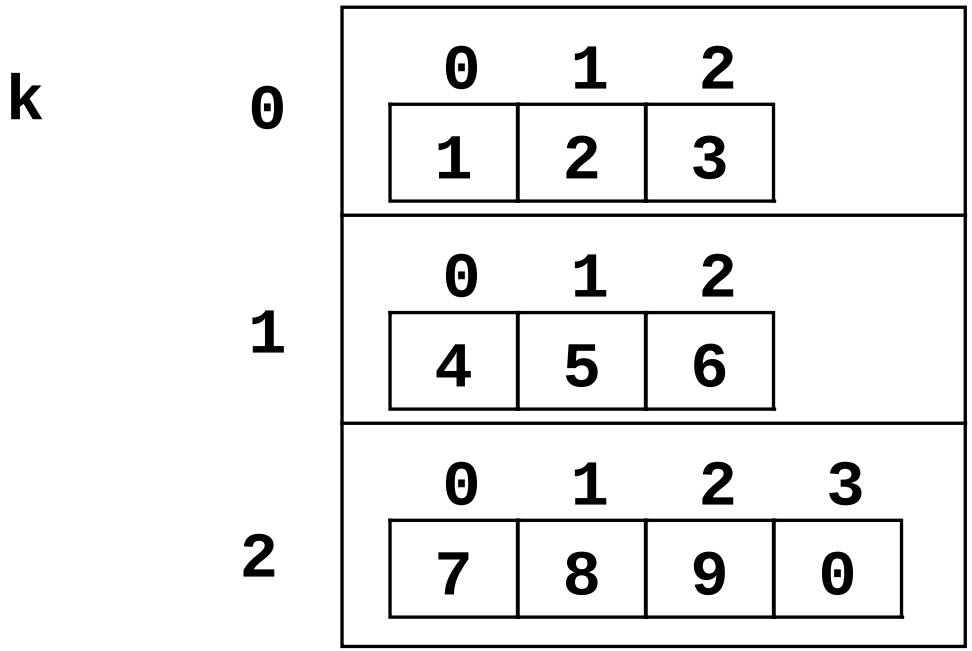
Príklady:

```
String[] mesiace = new String[]
   { "Jan","Feb","Mrc","Apr","Maj","Jun",
     "Jul","Aug","Sep","Okt","Nov","Dec" };
```



Príklady (Java) (pokrač.)

```
int[][] k = new int[][]
  { { 1, 2, 3 }, {4, 5, 6}, {7, 8, 9, 0} };
```



Príklady (Java) (pokrač.)

```
double[] x = new double[100];
```

Pole so 100 **`double`** prvkami, pričom každý prvok má počiatočnú hodnotu **`0.0`**; prvky sú inicializované implicitne

Pozor:

Prvky majú index od 0 po 99!

Príklady (Java) (pokrač.)

```
double[][] matrix = new double[50][100];
```

Dvojdimenzionálne pole (Matica) s **double prvkami**; pole poyostáva s 50 –tich polí, každé po 100 prvkov typu **double**, pričom kaˇydý prvok má hodnotu **0.0**;

Pozor:
matrix[0][0], ..., matrix[49][99]

Príklady (Java) (pokrač.)

```
double[][] matrix;
...
matrix = new double[50][100];
...
matrix = new double[][]
  {{0.0, 0.7}, {0.25, 0.0}, {0.0} };
```

Premenným typu polí môže byť priradená hocikedy hodnota zodpovedajúceho typu, t.j. musí zodpovedať počet a typ dimenzií

Indexy v poliach začínajú 0!

- V Jave začína indexovanie prvkov poľa pevne a to indexom 0
- Prvky sú vždy oindexované zasebou idúcimi číslami

Implicitná inicializácia (Java)

- S pomocou **`new`** vytvoríme nové pole; ak hodnoty poľa nie sú pri inicializácii explcitne uvedené, ale je uvedený iba počet prvkov, nadobúdajú prvky nasledovné implicitné hodnoty
  - **`0`** (resp. **`0.0`**) pri všetkých číselných typoch
  - **`false`** pri type **`boolean`**
  - **`‘\u0000‘`** pri type **`char`**
  - **`null`** inak (bude neskôr vysvetlené)

7.3 Prístup k prvkom poľa

Prvok poľa sa správa presne ako premenná daného typu

- **`matrix[0][1]`** má typ **`double`**
- **`matrix[0]`** má typ **`double[]`**
- **`vektor`** má typ **`double[]`**
- **`vektor[7]`** má typ **`double`**

Prístup k prvkom poľa

- Čítanie hodnoty vo výrazoch:

```
double erg = matrix[3][4] * vektor[2];
```

- Priradenie hodnoty prvku poľa:

```
matrix[3][4] = vektor[2] + erg;
matrix[2] = vektor;  //!!! Neskôr viac
```

Počítanie s indexom

- Index môže byť ľubovoľný výraz typu **int** (t.j. nielen konštanty ako doposiaľ)

Príklad:

```
double vysledok = 0.0;
for (int i = 0; i < vektor.length; i++)
{ vysledok = vysledok + vektor[i] * vektor[i]; }
System.out.println(“l= “ + Math.sqrt(vysledok));
```

Bubble Sort

```
// BubbleSort, Algoritmus na usporiadanie poľa.
public class BubbleSort
{
  public static void main (String[] args)
  {
    int[] a = new int[args.length];
    for ( int i = 0; i < args.length; i++ )
    { a[i] = Integer.parseInt(args[i]); }
```

Bubble Sort (Pokrač.)

```
boolean sorted;
do
{ sorted = true;
  for ( int i = 0; i < a.length - 1; i++ )
  { if ( a[i] > a[i+1] )
    { int h  = a[i];
      a[i]   = a[i+1];
      a[i+1] = h;
      sorted = false;
  } }
} while ( !sorted );
```

Pozor!

```
a[i] = a[i+1];
a[i+1] = a[i];
```

Nevymení prvky

a[i] a a[i+1]

Bubble Sort (Pokrač.)

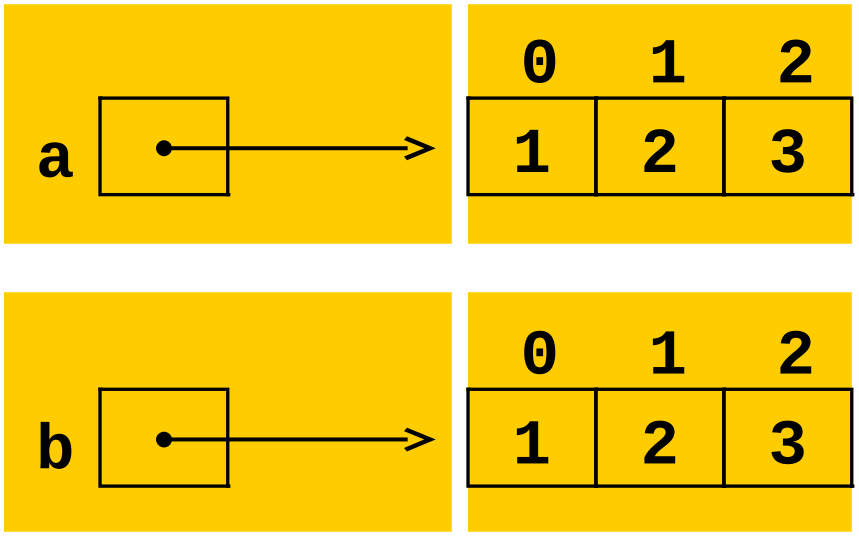
```
    System.out.println(
      "Cisla vo od najmensieho po najvacsie:");
    for ( int i = 0; i < a.length; i++ )
    { System.out.println(a[i]); }
  }
}
```

7.4 Polia a pointery

- S **new** vytvoríme nové pole daného typu
- Pri priradení takto vytvoreného poľa na premennú sa vloží do premennej adresa vytvoreného poľa, teda premenná bude ukazovať na pole.
- Pri priradení poľa na do premennej teda pole nie je kopírované!

Príklad 1

```
int[] a = new int[] {1, 2, 3};
int[] b = new int[] {1, 2, 3};
```

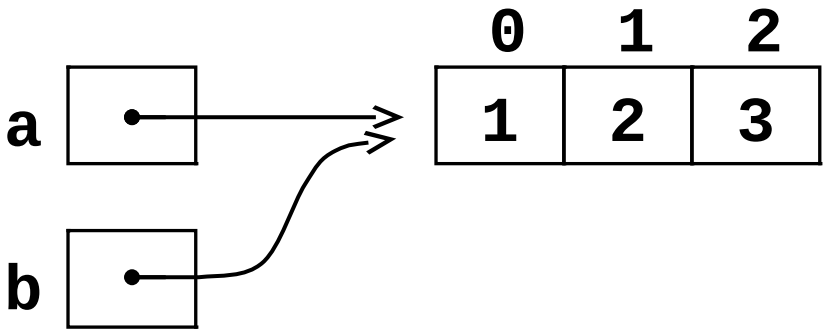


Príklad 2

```
int[] a, b;

a = new int[] {1, 2, 3};

b = a;
```



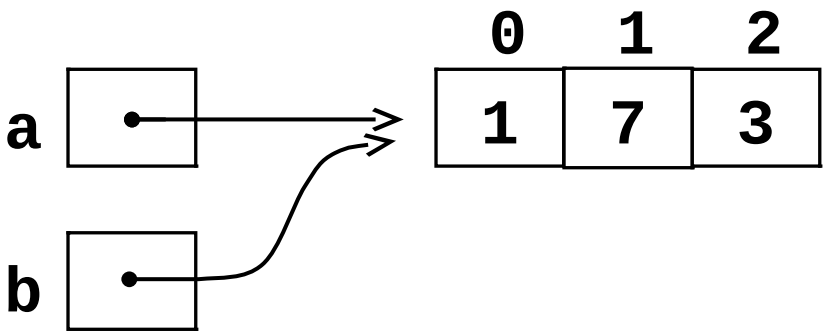
Pozor!

Pole nie je pri priradení kopírované!

Príklad 2 (Pokrač.)

```
a[1] = 7;

// Teraz platí b[1] = 7 !!!
```

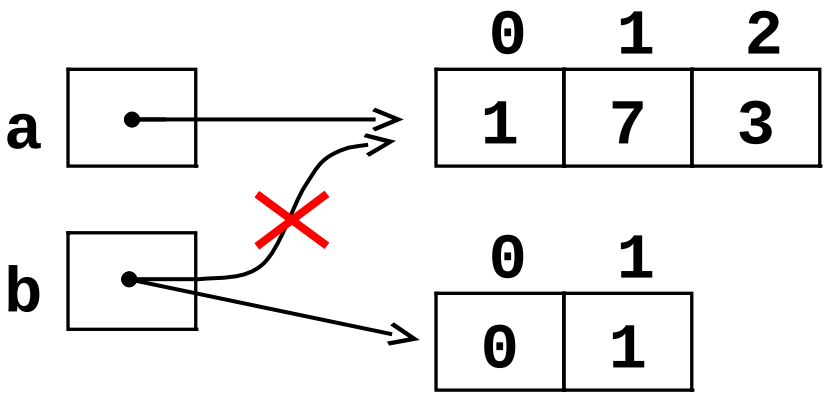


Pozor!

Priradenie hodnoty prvku a[1] zmenilo hodnotu prvku b[1], keďže obidve premenné a aj b ukazujú na to isté pole.

Príklad 2 (Pokrač.)

```
b = new int[] { 0, 1 };
```



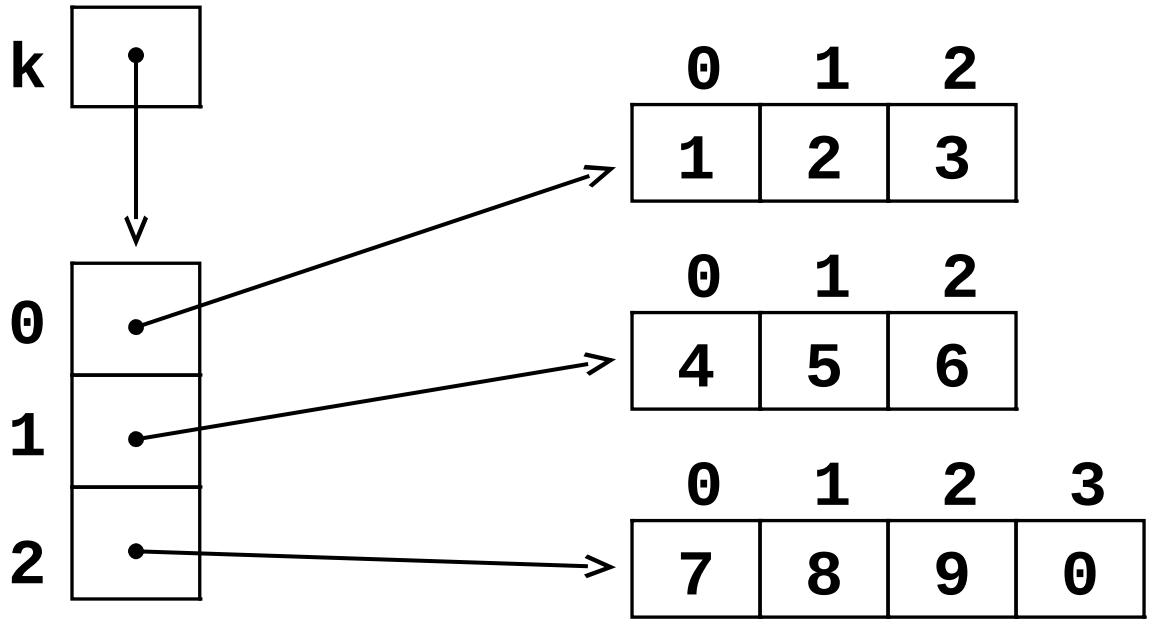
Pozor!

Pri priradení do premennej b sa samozrejme a nemení!

Premenná b bude totiž ukazovať na iné pole ako premenná a.

Príklad 3 Viacdimenzionálne polia

```
int[][] k = new int[][]
  { { 1, 2, 3 }, {4, 5, 6}, {7, 8, 9, 0} };
```



Príklad 3 (Pokrač.)

```
k[1] = a;
```