II. Základné elementy

8. Funkcie a procedúry

8.1 Motivácia

- Vo veľkých programoch sa často používa rovnaká časť programu viackrát
  (napr. výpočet odmocniny alebo usporiadanie poľa, ale aj výpis na obrazovku a pod.)
- **Funkcie** a **procedúry** umožňujú opakované použitie časti programu.

Príklad: Binomialkoeficient

```
public class Binomialkoeficient
{
  static int fak(int n)
  { int vysledok = 1;
    for (int i = 1; i <= n; i++)
    { vysledok = vysledok * i; }

    return vysledok;
  }
```

Definícia funkcie

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Príklad: Binomialkoeficient

```
  public static void main (String[] args)
  { int n = Integer.parseInt(args[0]);
    int k = Integer.parseInt(args[1]);

    int vysledok = fak(n) / ( fak(k) * fak(n – k) );

    System.out.println(„Vysledok: “ + vysledok);
  }
}
```

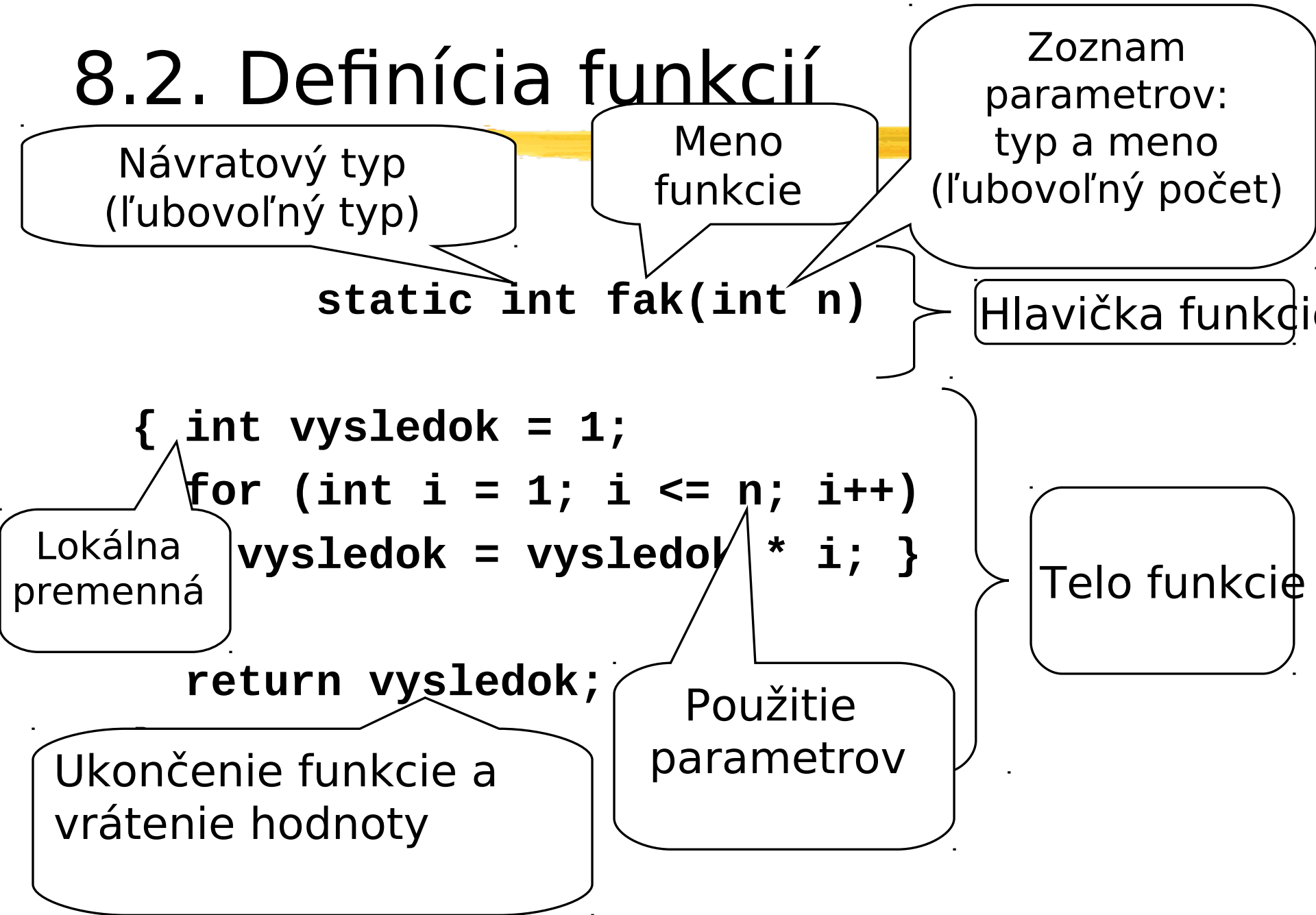
Volania funkcie

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Motivácia

- Vlastné funkcie môžeme používať rovnako ako preddefinované funkcie (napr. **`Math.abs`**, **`Math.sqrt`**, **`Math.max`**, ...).
- Použitím funkcií sa stáva program predľadnejším

8.2. Definícia funkcií



Hlavička funkcie

```
static int fak(int n)
```

- Definuje signatúru (Typy parametrov a typ návratovej hodnoty) funkcie. V našom príklade:

```
fak: int -> int
```

- **Typ návratovej hodnoty** a **typy parametrov** môžu byť ľubovoľné typy (napr. aj polia a pointery)
- Funkcia môže mať ľubovoľný počet parametrov, napr. aj žiadny: **`f()`**

Telo funkcie

- Telo funkcie je blok príkazov, ktorý sa vykoná pri zavolaní funkcie.
- V tele funkcie je možné použiť parametre funkcie. Parametre sa používajú rovnako ako premenné.
- Pomocou príkazu **`return`** sa ukončí funkcia a hodnota výrazu za kľúčovým slovom **`return`** je vrátená naspäť ako výsledná hodnota funkcie.

return-príkaz

- Výraz po kľučovom slove **return** musí mať rovnaký typ ako návratový typ funkcie daný v hlavičke.
- Príkaz **return** nemusí byť na konci tela funkcie.
- V tele funkcie môže byť príkaz **return** použitý viackrát.
- Akonáhle sa vykoná príkaz **return** je funkcia ukončená.

8.3 Volanie funkcie (Použitie funkcie)

- Funkcia môže byť použitá vo výrazoch. Pri volaní musí byť každému parametru funkcie (**formálne parametre**) priradený výraz zodpovedajúceho typu. Jeho hodnota sa stáva **aktuálnym parametrom**.
  Príklad: **`fak(n-k)`** volanie funkcie fak, kde formálny parameter n funkcie fak nadobudne hodnotu výrazu n-k (kde n a k sú premenné funkcie main)
- Zavolaná funkcia má vo výraze svoj návratový typ a hodnotu danú výrazom za príkazom return.

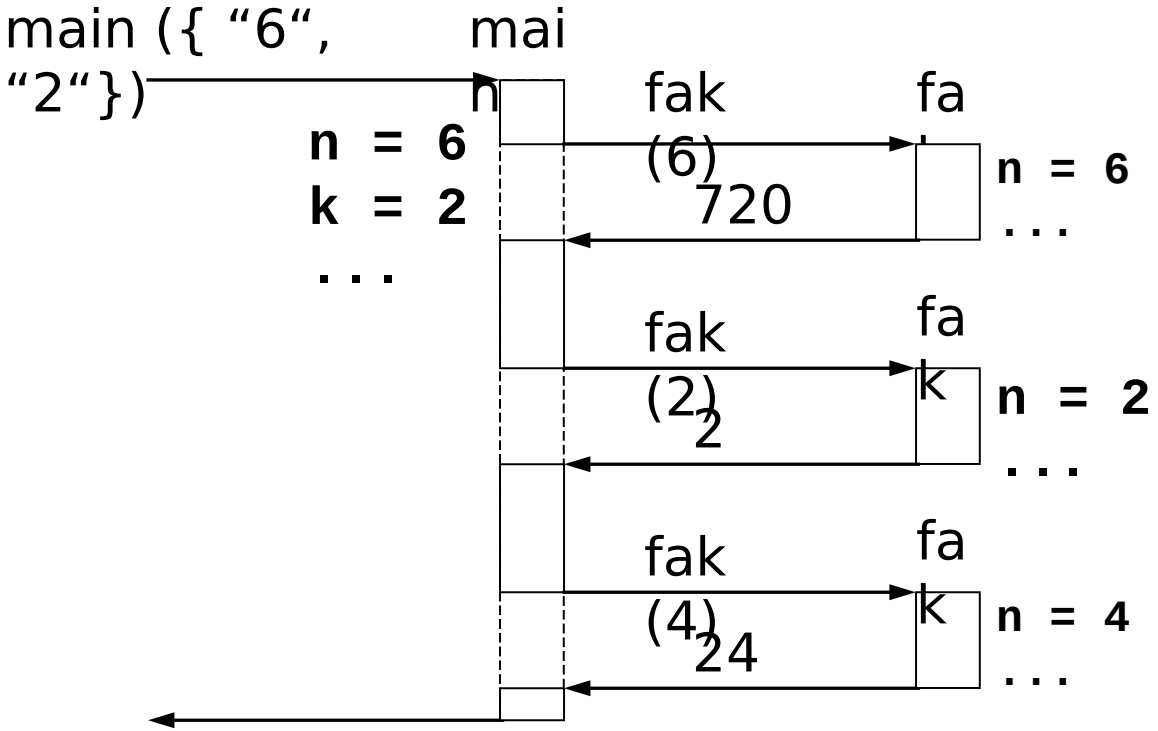
Príklad

```
static int fak(int n)
 { int vysledok = 1;
   for (int i = 1; i <= n; i++)
   { vysledok = vysledok * i; }
   return vysledok; }

 public static void main (String[] args)
 { int n = Integer.parseInt(args[0]);
   int k = Integer.parseInt(args[1]);
   int vysledok = fak(n) / ( fak(k) * fak(n - k) );
   System.out.println(„Vysledok: " + vysledok); }
```

Príklad (Pokrač.)

>Binomialkoeficient 6 2



8.4 Procedúry

- Procedúry sú funkcie, ktorých návratový typ je prázdny t.j. **`void`** (z angl. pre prázdny).
- Keďže procedúra nedáva žiadny výsledok, nemôže byť použitá vo výraze.
- Napriek tomu sú procedúry užitočné: **`System.out.println(“Bla“);`**

Zhrnutie

- Premenné typu pole sú pointery (ich hodnoty sú adresy polí)
- Funkcie
  - Hlavička funkcie
  - Telo funkcie
  - Volanie funkcie
  - Formálne vs. aktuálne parametre
- Procedúry (Návratový typ: **`void`**)

Bubble Sort (druhý pokus)

```
public class BubbleSort
{ public static void sort (int[] a)
  { 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 );    }
```

Bubble Sort (druhý pokus)

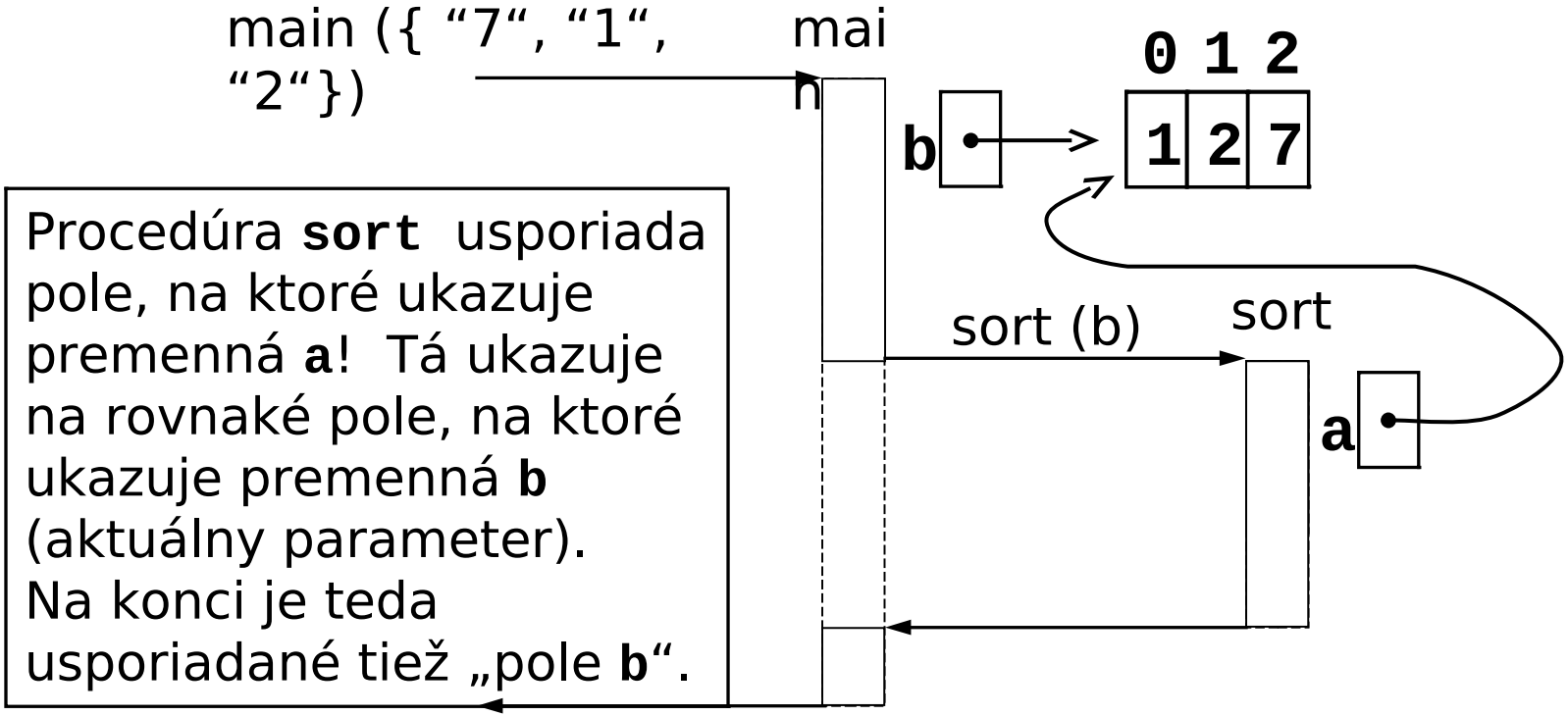
```
  public static void main (String[] args)
  { int[] b = new int[args.length];
    for ( int i = 0; i < args.length; i++ )
    { b[i] = Integer.parseInt(args[i]); }

    sort(b);

    System.out.println(„Zoradené pole:");
    for ( int i = 0; i < b.length; i++ )
    { System.out.println(b[i]); }
  }
}
```

Bubble Sort (druhý pokus)

>BubbleSort 7 1 2



Polia ako parametre funkcií

- Parameter typu pole (prvkov nejakého typu) je vlastne adresa poľa.
- Do parametra typu pole je teda uložená adresa poľa; t.j. pole nie je skopírované (funkcia pracuje s originálnym poľom).
- Ak teda procedúra (alebo všeobecne funkcia) modifikuje prvky poľa, je modifikované pôvodné pole

Pozorovanie:

```
public static void sort (int[] a)
{ 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 );
}
```

Tu nemusí byť **`return;`**

Príkaz **`return`** v procedúrach

- Na konci tela procedúry nemusí byť príkaz **`return`**; procedúra je na konci automaticky ukončená.
  (V prípade funkcií s neprázdnym návratovým typom toto nie je možné, keďže takéto funkcie musia vrátiť nejakú hodnotu)

- V procedúrach tiež môžeme použiť príkaz **`return`** . V takom prípade za ním nenasleduje žiaden výraz.

8.5 Volanie hodnotou a volanie adresou

- Pri volaní funkcie je vždy kopírovaná do formálneho parametra hodnota. Ak je formálny dátový typ primitívny, tak takéto volanie nazývame volanie hodnotou (call by value);
- Pri neprimitívnych dátových typoch, napr. pri premenných typu pole, čo sú vlastne premenné, ktorých hodnotou je adresa objektu (napr. poľa) hovoríme o volaní adresou (call by name)
- Ak pri volaní hodnotou zmeníme formálny parameter, hodnota premennej použitej ako aktuálny parameter sa nezmení. Ak pri volaní adresou zmeníme hodnotu objektu, na ktorý ukazuje formálny parameter, zmeníme zároveň hodnotu objektu, na ktorý ukazuje aktuálny parameter. Je to logické, keďže sme odovzdali adresu a teda formálny aj aktuálny parameter ukazujú na to isté.

Volanie adresou a postranné efekty

- Pomocou volania adresou môže funkcia alebo procedúra „vrátiť“ nové hodnoty (t.j. zmeniť hodnoty) (pozri. Bubble Sort)!

- Pomocou volania hodnotou to nie je možné.

Polymorphismus

- Je tiež možné definovať funkcie s rovnakým menom a s rozdielnym počtom a typom parametrov.
- Ktorá funkcia má byť zavolaná je dané typom a počtom aktuálnych parametrov.