# 17. Datová struktura pole

Skupina proměnných obvykle stejného typu, které jsou v paměti alokovány za sebou. K těmto proměnným lze přistupovat pomocí jejich indexů (pořadí v této posloupnosti, počítáno od 0). Z tohoto způsobu alokace je zřejmé, že pole má fixní délku, kterou nelze nijak změnit (protože paměť v oblasti za polem může být obsazena jiným objektem).

## Dělení Polí

* Statické
  + Velikost je jistá před překladem, interpretací programu, obvykle indexované od 0
* Dynamické
  + Velikost závisí na průběhu programu, a lze měnit i po jeho vytvoření
  + Pošle se požadavek na správce paměti a ten buď přidělí paměť, nebo oznámí chybu
  + V Javě nelze měnit velikost pole za běhu programu
* Indexované
  + Přístup k prvku pomocí čísla, většinou od 0
  + Java má přístup k prvkům pouze pomocí indexu
* Asociativní
  + Přístup pomocí řetězce
  + V PHP lze přistupovat pomocí řetězce
* Homogenní
  + V poli může být jen jeden datový typ
* Různorodé
  + V poli můžou být různé datové typy
  + V Javě pole nemůže být různorodé
  + V PHP pole může být různorodé

## Vícerozměrná pole

Vícerozměrná pole v Javě jsou ve skutečnosti jen pole plné polí.

Při deklaraci vícerozměrných polí se postupuje obdobně jako u pole jednorozměrného, jediným rozdílem je počet závorek, který se uvádí.

Dvojrozměrné pole je časté, trojrozměrné je neobvyklé a více jak trojrozměrné je obvykle chyba návrhu aplikace (jenom málokdo si umí taková data představit).

#### Typ[][] jmeno = new typ[pocet\_prvku][pocet\_prvku];

## JAVA

V okamžiku, kdy se pole vytvoří, tak jsou jeho hodnoty přednastaveny do výchozího stavu. V případě integerů 0, u booleanu false, a v případě referencí ukazatel do prázdna null.

#### typ[] jmeno;

Vytvoří se proměnná pole, ale zatím se nealokuje místo v paměti.

Místo v paměti se alokuje konstruktorem.

#### jmeno = new typ[velikost\_pole];

Inicializaci a deklaraci pole lze zapsat obdobně jako u ostatních typů.

#### typ[] jmeno = {hodnota, hodnota2, hodnota3, hodnota4};

#### typ[] jmeno = new typ[velikost pole];

## PHP

V PHP existuje spoustu syntaxí pro vytvoření pole.

#### $pole = array("test", 2, 2.16);

#### $pole1 = array(0 => "test", 1 => 2, 2 => 2.16);

#### $pole2 = array("pozice" => "test", "pozice2" => 2, "pozice3" => 3.14);

#### $pole3 = ["test", 2, 2.16];

#### $pole4 = [0 => "test", 1 => 2, 2 => 2.16];

#### $pole5 = ["pozice" => "test", "pozice2" => 2, "pozice3" => 3.14];

Jak je vidět na příkladech, pole v PHP je asociativní a různorodé.

V PHP lze vytvořit i vícerozměrné pole.

#### $a1 = array("a" => 0, "b" => 1);

#### $a2 = array("aa" => 00, "bb" => 11);

#### $together = array($a1, $a2);

## VBA

#### **Dim** *nazev(***velikost***)* **As** *Datovy\_typ*

### Metody nad třídou Array:

Třída Array obsahuje pomocné metody pro práci s poli.

* Sort
  + Metoda pro třídění pole
* Reverse
  + Obrácení posloupnosti
* IndexOf, LastIndexOf
  + Tyto metody vrátí index prvního nebo posledního nalezeného prvku
* Copy
  + Zkopíruje část pole do jiného pole
  + Prvním parametrem je zdrojové pole, druhým cílové a třetím počet znaků, který se má zkopírovat

### Metody nad polem:

Třída Array není jedinou možností, jak s polem manipulovat. Přímo na samotné instanci pole (konkrétní proměnné) lze volat také spoustu metod.

* Length
  + Vrátí délku pole
* Min, max, average, sum
  + Vracejí minimum, maximum, průměr, součet všech prvků
  + Bez parametrů
* Concat, intersect, union
  + Vrátí na výstupu nové pole
  + Jako parametr mají druhé pole
  + Spojení; Průnik
  + Sjednocení; prvky, které byly v obou, jsou zde pouze jednou
* First, last
  + Vrátí první, poslední prvek
* Take, skip
  + Berou jako parametr počet prvků
  + Take vrátí pole s daným počtem prvků zkopírovaných od začátku
  + Skip naopak vrátí pole bez těchto prvních prvků
* Contains
  + Vrací True/False podle toho, zda se prvek v parametru metody, nachází v poli
* Reverse
  + Vrátí nově vytvořené pole s otočenými prvky
* Distinct
  + Zajistí, aby byl v poli každý prvek jen jednou
  + Vrátí nově vytvořené pole