4. Datové typy (C, Java)

**Datový typ určuje pro proměnou**:

* Množinu přípustných hodnot
* Množinu operací
* Velikost obsazené paměti

### **Lokální proměnné**

* Definovány uvnitř těla funkce/bloku
* Po konci bloku proměnné zanikají
* Ukládány v Zásobníku (stack)
* Výjimkou třída static

### **Globální proměnné**

* Přístupné od deklarace a zanikají při konci souboru
* Automaticky inicializovány na 0
* Přístupné kdekoliv v programu
* Ukládány do Datové oblasti

## **Jazyk C**

* Striktně typový jazyk
* Nutno uvést název ale i typ proměnné

1. **Jednoduché datové typy (jedna hodnota)**
2. **Celočíselné (ordinální - lze seřadit)**
   * + mají předchůdce a následovníkaObsah obrázku text, snímek obrazovky, displej, Písmo

       Popis byl vytvořen automaticky
     + signed / unsigned
   * short (16b)
   * int (16/32b dle typu CPU)
   * long/long int (32b/64b dle typu CPU)
   * char (8b)
   * enum
     + výčtový datový typ
     + číselník
3. **Reálné**
   * float (32b) – 6 desetin. míst
   * long double/double long (64b) – 15 desetinn. míst
   * double (80b) – 19 destinn. míst
4. **Speciální**
   * Void (bez hodnoty, když hodnota není podstatná)
   * Pointer
     + Hodnotou je adresa
     + Spjaty s daným typem proměnné
5. **Strukturované datové typy (více hodnota)**
6. **Homogenní**
   * Prvky stejného datového typu
   * Pole a řetězce (řetězce zakončeny ukončovací nulou ‘\0‘)
   * Soubor (FILE)
     + mimo operační paměť pc - na disku
     + textový nebo binární
7. **Heterogenní**
   * Prvky různého datového typu
   * Struktura (podobné třídám v OOP) -zabírá paměť jako celek
     + záleží jak je vytvořena v paměti (dvojí přístup - staticky, přes pointery)
   * Union (podobné struktuře) – zabírá paměť podle největší položkyObsah obrázku text, snímek obrazovky, displej, Písmo

     Popis byl vytvořen automaticky

Obsah obrázku text, řada/pruh, Písmo, snímek obrazovky

Popis byl vytvořen automaticky

**Přístup ke členům struktury**

**struct Osoba {**

**int vek;**

**int pocet\_pratel;**

**};**

**struct Osoba martina = {18, 10};**

**Statické struktury: <výraz typu struktura>.<název členu>**

martina.vek += 1; // přístup k členu `vek`

**Pomocí pointerů: <pointer na strukturu>-><název členu>**

martina->pocet\_pratel++;

* lze zapsat i : (\*martina).pocet\_pratel++;
* pozor na prioritu (**\*** menší priorita než **.**) - nutno ozávorkovat

Operátor **->** je čistě syntaktickou zkratkou

## **Jazyk Java**

* Vychází z C
* Datové typy se liší obvykle jen velikostí
* OOP

1. **Primitivní datové typy (uložena pouze hodnota)**
   * Reprezentují elementární údaje
   * Nedají se uživatelsky definovat
   * Celočíselné
   * Reálné
   * Znakové
   * Logické (boolean pouze jako true/false a ne 1/0)
   * Void

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

1. **Referenční datové typy**
   * Třídy
   * Hodnota referenční proměnné = odkaz na místo uložení v paměti
   * V C využívány pointery
   * vlastnosti a metody

Obsah obrázku text, Písmo, řada/pruh, snímek obrazovky

Popis byl vytvořen automaticky

## **Rozdíl mezi C a Java**

**Pole v C**: int pole[10];

**Pole v Javě:** int[] pole = new int[10];