Datové typy a struktury

Jednoduché datové typy

- **Boolean** = logická hodnota (true / false)
  - K uložení stačí **1 bit** – často celé slovo (1 byte)

- **Character** = **znak**
  - Pro 8-bitový ASCII kód stačí **1 byte** (256 možností)
  - Pro UNICODE jsou potřeba **2 byty** (65 tis. možností)

- **Integer** = číslo v pevné řádové čárce
  - Pro uložení **2 byty** $x \in <-32768, 32767>$
  - **Longint** uložen ve **4 bytech** $x \in <-2^{31}, 2^{31}-1>$

- **Real** = číslo v pohyblivé řádové čárce
  - **IEEE formát** uložen ve **4 bytech**
  - **Formát Double - 8 bytů**

- **Ordinální datový typ**
  - hodnoty typu lze opatřit **pořadovými (ordinálními) čísly**
  - pořadová čísla jsou základem porovnání hodnot
    - následník - hodnota s ordinálním čísle o 1 větší
    - předchůdce - hodnota s ordinálním čísle o 1 menší
  - **Ordinální typy:**
    - **integer** – každá hodnota je zároveň ordinálním číslem
    - **character** – ordinální čísla jsou dána kódováním (ASCII)
    - **boolean** – false=0, true=1

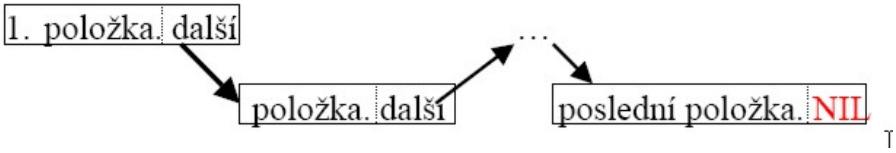
Složené (strukturované) datové typy

- Získáme **z jednoduchých datových typů**, ale i ze složených hierarchickým postupem

- **Homogenní** (položky jsou stejného druhu)

  - **S přímým přístupem** – **Pole - Array**
    - Jednotlivé položky se identifikují svým ***indexem*** $A_1, A_2, \ldots, A_N$ nebo A[1], A[2], …, A[N]
    - Rozsah N nutno znát předem (nebo nechat reservu a sledovat kam až je platně naplněno)
    - **Pole znaků** = *znakový řetězec* - **string**

- **S postupným (sekvenčním) přístupem - *Proud – Streem***
  - **V paměti obvykle seznam – list**

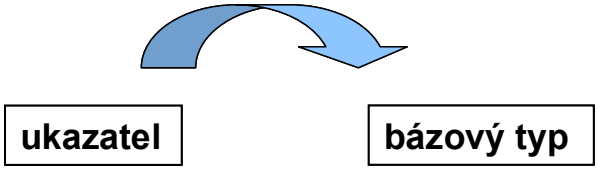


  - **Na vnějším zařízení** jako **soubor – file**

  - **Datový typ množina – set**
    - množina prvků některého ordinálního typu – **bázový typ**
    - má-li bázový typ **N prvků**, existuje 2^N možných **podmnožin** tohoto bázového typu
    - každá **podmnožina** popsána jednoznačně **N-ticí nul a jedniček**
    - **hodnota 1** je užita v případě, že **prvek** bázového typu daného pořadí **je přítomen**, **hodnota 0** v případě, že **není přítomen**
    - pro **každý prvek** bázového typu je třeba **1 bit**
    - pro množinu jsou definovány operace **sjednocení**, **průnik**, **rozdíl**, **podmnožina**, **prvek množiny**

- **Nehomogenní** (položky různých druhů)
  - **Záznam - Record**
    Příklad: Zboží:
    - kód zboží – *Integer*
    - název zboží – *String (pole znaků)*
    - cena zboží – *Real*
    - daň zaplacena ANO/NE – *Boolean*
  - **Objekt – Object**
    - rozšíření typu **záznam**
    - obsahuje navíc **metody** zpracování dat
    - DPH = cena zboží * 0,19 – *Real*

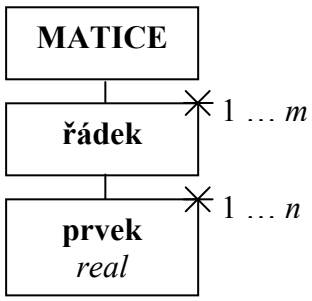
- **Ukazatel – pointer**
  - vždy ukazuje na jiný datový typ (**bázový typ**)
  - obsahuje adresu, na které je uložena hodnota bázového typu
  - může obsahovat i speciální hodnotu **nil** = *ukazatel nikam neukazuje*



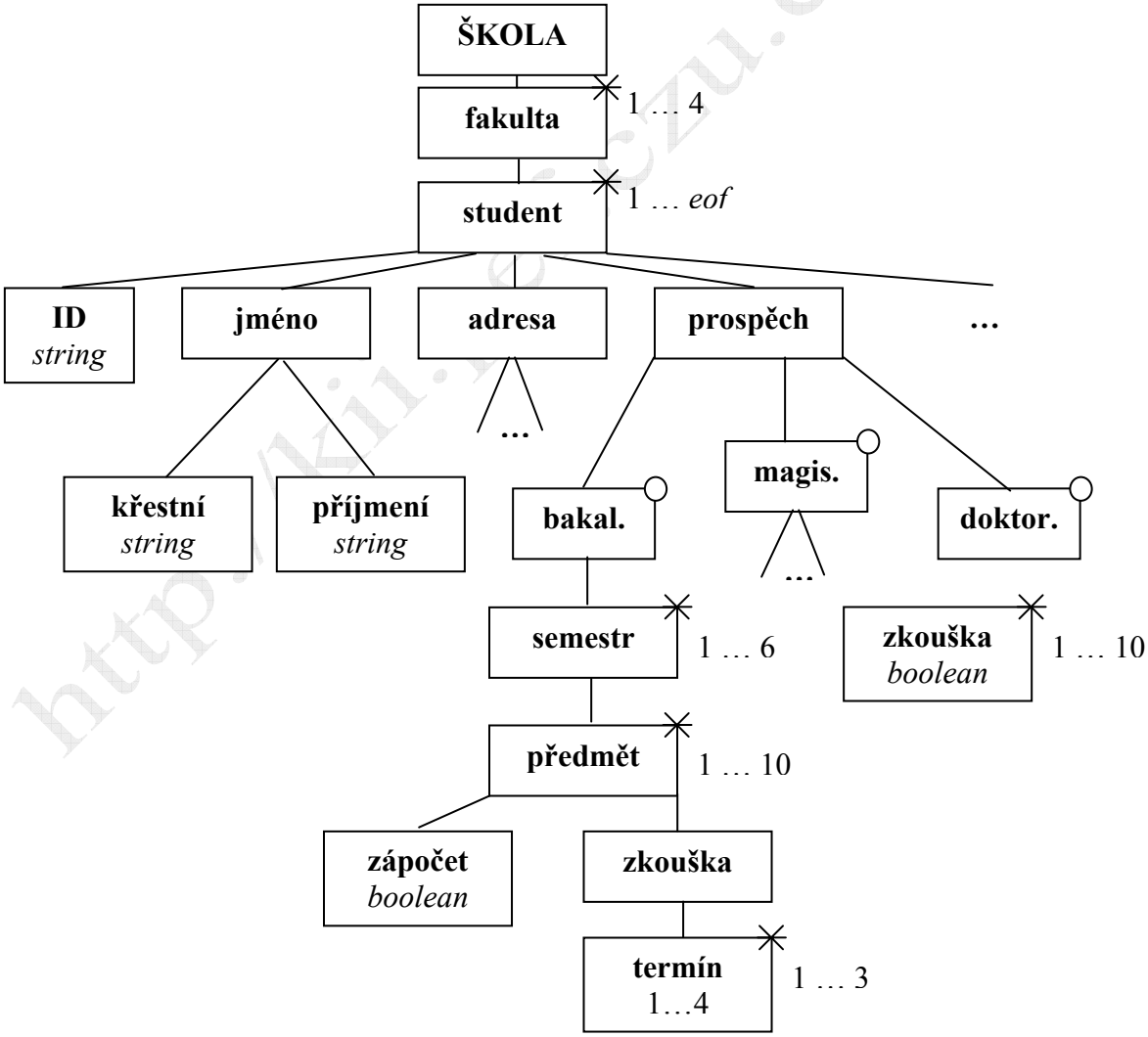
Hierarchie složených datových typů

- složené datové typy je účelné vytvářet na **více** (i mnoha) **hierarchických úrovních**
- **rozklad složeného typu** je vhodné znázornit graficky jako **strom**
  - pro **homogenní rozklad** (opakování téhož typu) je zvykem použít symbol *****
  - pro **výběr z několika možností** se užívá značka **O**

- **Příklady:**
  - Matice reálných čísel typu ***m* · *n***



  - Soubor základních údajů o studentech university

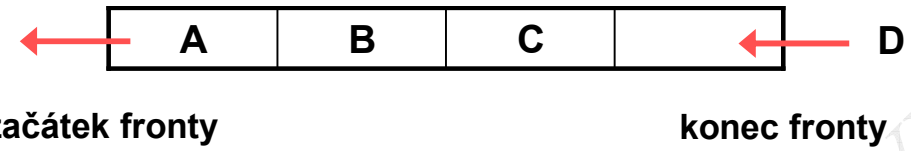


- Pečlivá analýza struktury dat je nezbytná pro **návrh programu**, který je má zpracovávat.

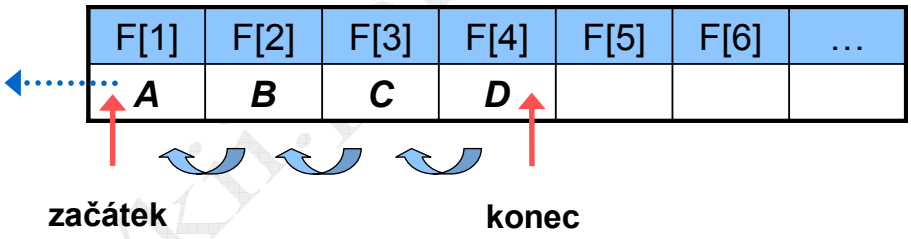
Speciální datové struktury

Fronta

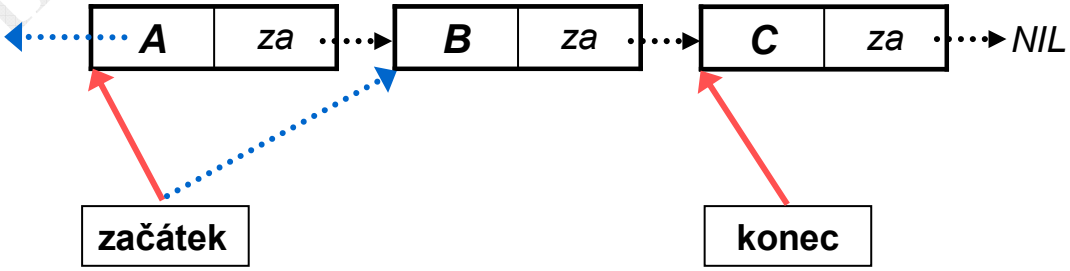
- řada prvků, které jsou na jeden konec fronty přidávány a z druhého konce odebírány
- přístup **FIFO** (*First In First Out*)



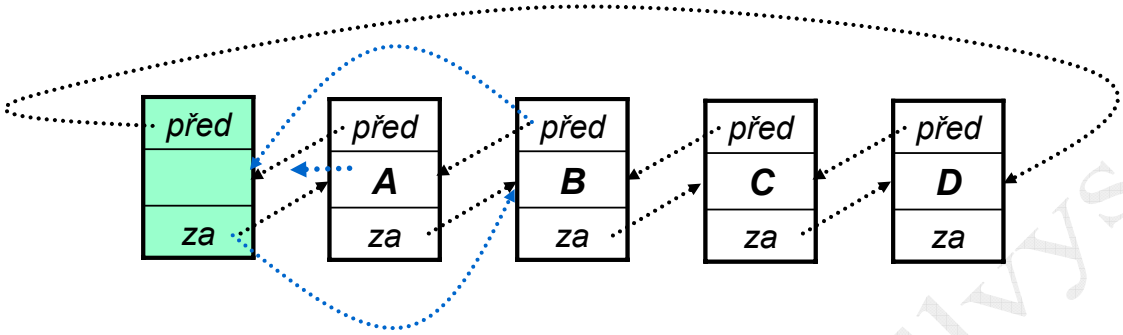
- **Použití:**
  - požadavky na vstupní a výstupní zařízení v OS
  - vyrovnávací a synchronizační paměti (***buffer***)
  - simulační systémy
- **Implementace** – vyžaduje rychlý přístup na oba konce
  - **pole prvků**
    - začátek fronty = **první index**
    - konec fronty = **index posledního prvku**
    - při odebrání prvku ze začátku fronty nutný posun ostatních prvků o jedno místo



  - **jednosměrný seznam**
    - **začátek** = ukazatel na první prvek seznamu
    - **konec** = ukazatel na poslední prvek seznamu
    - při odebrání prvku z fronty se ukazatel **začátek** přesune na další prvek seznamu

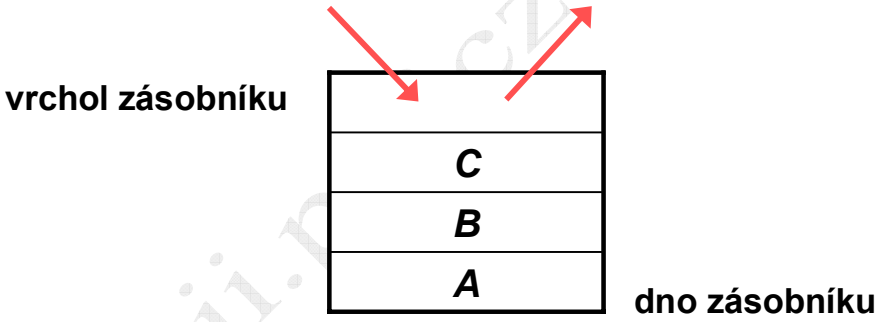


- **dvousměrný cyklický seznam**
  - navíc jeden prvek seznamu = **hlava**
  - **hlava** je **začátkem** i **koncem** fronty
  - při vyjmutí prvku z fronty pouze změna ukazatele v prvku hlava

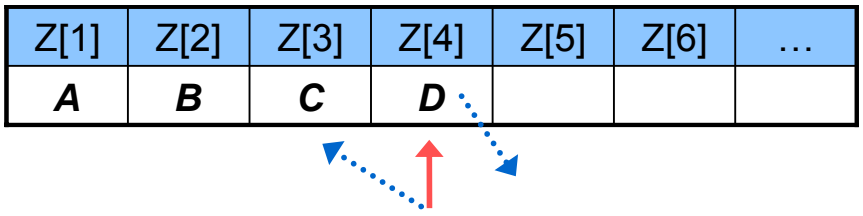


Zásobník

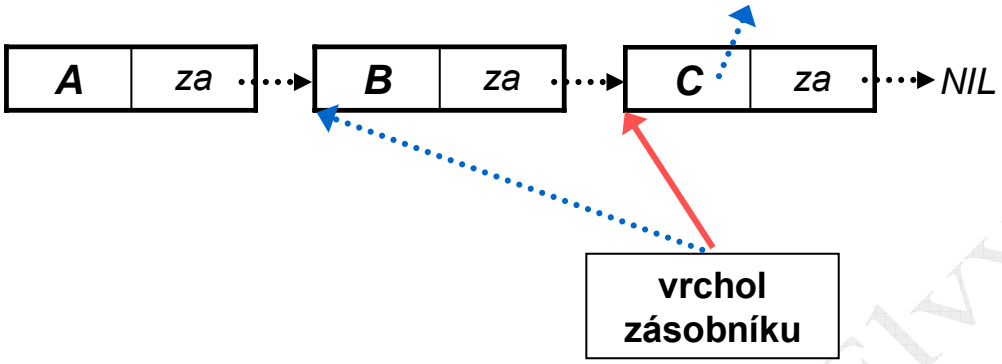
- množina prvků, které jsou přidávány i odebírány z téhož konce = **vrchol zásobníku**
- přístup **LIFO** (*Last In First Out*)



- **Použití:**
  - při volání podprogramu – uložení návratové adresy
  - ukládání lokálních proměnných
- **Implementace** – vyžaduje přístup jen na jeden konec
  - **pole prvků**
    - vrchol zásobníku = **index** posledního prvku pole
    - při odebrání prvku z pole se vrcholem stává předcházející index



- **jednosměrný seznam**
  - **vrchol zásobníku** = ukazatel na poslední prvek seznamu
  - při odebrání prvku ze zásobníku se ukazatel na vrchol zásobníku přesune na předcházející prvek



Graf

- **Graf** (orientovaný, neorientovaný) se skládá z uzlů a hran
  - **uzly** jsou prvky nesoucí informaci a odkazy na další uzly
  - **hrany** (reprezentovány ukazateli) určují vazby mezi uzly

- **Strom** je souvislý orientovaný graf bez cyklů (v neorientovaném stromu lze za kořen prohlásit libovolný vrchol → tím je určena orientace)
  - každý uzel má jednoho předchůdce a jednoho nebo více následovníků
  - **kořen** = nemá žádného předchůdce
  - **list** = nemá žádné následovníky
  - **binární strom** = každý uzel **max. dva následovníky**

- **Implementace**
  - nejčastěji různé typy **seznamu**
  - např. **binární strom**