VŠB TECHNICKÁ UNIVERZITA OSTRAVA | HORNICKO GEOLOGICKÁ FAKULTA | KATEDRA GEOINFORMATIKY

Programování v GIS 1
2 – Datové typy a struktury

Michal Kačmařík

A924, tel.: 5512

e-mail: michal.kacmarik@vsb.cz

https://www.hgf.vsb.cz/548/cs/

Datové struktury

- Při běhu programu je nutno v paměti uchovávat data. Reprezentace dat je dána datovým typem (DT). Volba DT je řešena z hlediska efektivity zpracování dat.
- *Základní datové struktury* lze charakterizovat neměnností svého definovaného rozsahu.
- *Abstraktní datové struktury* – implementovány jako objekty, které za běhu programu svůj rozsah mohou měnit.

Datový typ

Datový typ (dále DT) vymezuje množinu hodnot, kterých může nabýt konstanta, proměnná, funkce nebo výraz a množinu operací nad těmito hodnotami.

Definice objektu (konstanty, proměnné, funkce) znamená:

- přiřadit jednoznačné jméno (identifikátor)
- a určit datový typ. Tím je vyhrazen prostor v paměti pro tento objekt (počet bajtů).

Pozor: v jazyce Python nedefinujeme datový typ proměnné, určí se automaticky dle jejího obsahu

Základní datové typy

Jednoduché datové typy

Boolean

Číselný DT

Znak

Základní datové struktury

Základní DT – Boolean

- Logický typ
- Binární typ
- Objekt datového typu *boolean* se používá v případě pouze dvou opačných hodnot - *nepravda* a *pravda*. Nad logickým datovým typem jsou definovány *logické operace*, kdy výsledkem je opět *logická hodnota*:
  - negace, (0 → 1, 1→ 0)
  - konjunkce (logický součin), (0 & 1)
  - disjunkce (logický součet) (1 ! 0).

- Deklarace:
  - *boolean boolT = true;*
  - *boolean boolF = false;*

Základní DT – Číslo

Číselný DT ***celé číslo (integer, …)*** je objekt, který nabývá hodnot z množiny celých čísel.

Objekt datového typu ***reálné číslo (float, double, …)*** nabývá hodnot z množiny reálných čísel.

V obou případech závisí *rozsah*, případně *přesnost* s jakou jsou čísla reprezentována, na konkrétním operačním systému a použitém překladači.

Nad číselným DT jsou definovány tyto operace:

- *aritmetické operace (+, -, /, *, %, …),*
- *relační operace*.

Standardní rozsahy číselných DT

typ	popis	velikost	min. hodnota	max. hodnota
byte	celé číslo	8 bitů	-128	127
short	celé číslo	16 bitů	-32768	32767
integer (int)	celé číslo	32 bitů	-2147483648	2147483647
long	celé číslo	64 bitů	-9223372036854780000	9223372036854780000
float	reálné číslo	32 bitů	-3.40282e+38	+3.40282e+38
double	reálné číslo	64 bitů	-1.79769e+308	+1.79769e+308

Některé jazyky rozlišují variantu **signed** (se znaménkem) nebo **unsigned** (bez znaménka, nezáporné). Tato vlastnost udává, jaké hodnoty může typ nabývat (tj. jestli mohou být i záporné nebo ne)

Deklarace číselných typů

Příklady deklarace proměnných číselného DT v jazyce C:

- *byte b = 100;*
- *short s = -30000;*
- *int i = 10000;*
- *long l = 100;*
- *double dA = 0.5;*
- *double dB = .5;*
- *double dC = 5E-1;*
- *float f = 100.5f*

Základní DT – znak (char)

- Množina hodnot DT znak je tvořena
  - *znaky abecedy* (malá, velká písmena),
  - *číslicemi*,
  - *speciálními znaky*.
- Každému znaku je přiřazena celočíselná hodnota, tzv. *kód znaku*. Ten je určen pořadím v tzv. kódovací tabulce - kódování podle norem ASCII, Unicode, apod.
- Například při řazení podle abecedy je pak využito použité kódování.

- Syntaxe v jazyce Python:
  - promenna = “a“ # bude povazovano za znak (text)
  - promenna = 1 # bude považováno za cele cislo
  - promenna = “1“ # bude povazovano za znak (text)
  - promenna = ‘1‘ # bude povazovano za znak (text)

Konverze datových typů

- Konverze = převod z jednoho DT na jiný
- Ukázky v jazyce Python:

```
print(type(promenna)) #na standardni vystup se vytiskne DT promenne


int(promenna) #konverze DT na celé číslo (integer), pozor, dojde jednoduse k
#useknuti hodnot za desetinnou carkou, napr.
print(int(1.6)) – vrátí 1


float(promenna)  #konverze DT na reálné cislo (float)
str(promenna)     #konverze DT na textovy reteze(string)
```

Struktura

- Struktura je tvořena několika prvky (položkami), obecně různého datového typu.
- Definice struktury znamená pojmenování struktury a určení datového typu jednotlivých položek a jejich pojmenování.
- Pomocí *identifikátorů položek* se v algoritmu přistupuje k hodnotě příslušné položky.

Základní datové struktury

- Složené datové typy: jeden nebo více prvků.
- Homogenní – prvky stejného DT:
  - Pole – Array
  - Řetězec - String
- Nehomogenní – prvky různého DT
  - Seznam – List
  - Slovník - Dictionary

Pole (array)

- Polem rozumíme posloupnost prvků stejného DT.
- primární operací nad DT pole je *přístup k jednotlivým prvkům* pole pomocí ***indexu***, tj. celého čísla, které udává pozici prvku v poli. Pole jsou proto setřízená (pořadí prvku v rámci pole je dáno).
- Prvkem pole může být opět pole – vznikají dvou a vícerozměrná pole.
- Speciálním, ale hojně používaným případem pole je ***řetězec (string)***, jehož prvky jsou typu znak.
  - promenna = “jmeno“
  - promenna = ‘jmeno‘

Pole (array)

Int [] prvocisla
String [] dnyvTydnu

[1, 2, 3, 7, 11, 17, 19]

["pondeli", " utery", " streda", " ctvrtek", "patek", "sobota", "nedele"]

[42.67, 27.89, -5.3, 7.43, 1.09]

[42.67, 27.89, -5.3, 7.43, 1.09
2.67, 2.8, -5.73, 10.12, 107.0]

[0,1,0,1,
0,0,1,1,
1,1,1,0]

Index

- Umožňuje přistupovat k jednomu prvku či k výčtu prvků v poli (a v jiných datových strukturách)
- Příklady v jazyce Python:

```
cars = ["Ford", "Volvo", "BMW"] #deklarace pole o trech prvcich
print(cars[0]) #na standardni vystup se vytiskne prvni prvek pole cars
```

POZOR: Python indexuje od 0!

```
print(cars[0:2]) #na standardni vystup se vytisknou první dva prvky pole cars
print(cars[-1])   #na standardni vystup se vytiskne poslední prvek pole cars
cars[0] = "Toyota" #prvni prvek v poli se zmeni na hodnotu “Toyota“
x = cars[1] #do promenne x se ulozi druhy prvek pole s nazvem cars
x = len(cars) #do promenne x se ulozi informace o poctu prvku v poli s nazvem cars
```

Ukazatel

- Ukazatel (pointer) neobsahuje přímo data uložená v proměnné, ale určuje pouze polohu této proměnné v paměti.
- Rozdíl mezi ukazatelem a indexem pole spočívá v tom, že index určuje polohu proměnné v poli – *i-tý prvek*, zatímco ukazatel obsahuje přímo *adresu buněk* paměti počítače, kde je proměnná uložena.
- Ukazatele se používají v programovacích jazycích pro práci s proměnnými, které vytváříme při běhu programu (často neznáme množství dat s nimiž budeme pracovat, proto na dynamicky vytvořenou proměnnou odkazujeme pomocí ukazatele).

Seznam (list)

- Seznam = pole, kde jednotlivé prvky mohou být různého datového typu.
- Viz dále v prezentaci
- Ukázka v jazyce Python:

  seznam = ["rohlik", "chleba", 5, "houska“, -59.99, "koblizek", 0.1]

- Jazyk Python má dvě varianty:

  seznam_list = ["rohlik", "chleba", 5, "houska“, -59.99, "koblizek", 0.1] – obsah seznamu může být měněn

  seznam_tuple = ("pondeli", "utery", "streda", "ctvrtek", "patek") – obsah seznamu nemůže být měněn

https://www.w3schools.com/python/python_lists.asp

Sada (set)

- Typ seznamu, který má specifické vlastnosti:
  - Je možné přidávat nové prvky, mazat existující prvky, není však možné editovat existující prvky
  - Sada je nesetřízená, prvky nemají indexy
  - Jedna sada nemůže obsahovat dva identické prvky

- Ukázka v jazyce Python:

```
sada = {"rohlik", "chleba", 5, "houska“, -59.99, "koblizek", 0.1}
```

https://www.w3schools.com/python/python_sets.asp

Slovník (dictionary)

- Umožňuje ukládat k sobě patřící dvojice hodnot (klíč:hodnota)
- Typické příklady:
  - telefonní seznam v podobě jmeno_prijmeni:telefon
  - seznam osob s datem narození v podobě jmeno_prijmeni:datum
  - Katalog výrobků v podobě vyrobek:cena
- Syntaxe v jazyce Python:

```
slovnik = {'michal':5512,'petr':5470,'igor':3550} #priklad telefonniho seznamu
print(slovnik.keys())  #funkce keys vraci klice slovniku (klic = prvni prvek dvojice)
print(slovnik.values()) #funkce values vraci hodnoty slovniku (value = druhy prvek dvojice)
print(slovnik.get('michal')) #funkce get vraci hodnotu pro zadany klic
```

https://www.w3schools.com/python/python_dictionaries.asp

Abstraktní datové typy

- Za abstraktní datový typ označujeme DT, který je na implementaci nezávislý a specifikuje strukturu dat a odpovídající operace nad touto strukturou.
- Souhrnně jsou označovány pojmem *kontejner*.
- Kontejner (kolekce) slouží k organizovanému skladování prvků podle určitých pravidel.

ADT – Kontejner

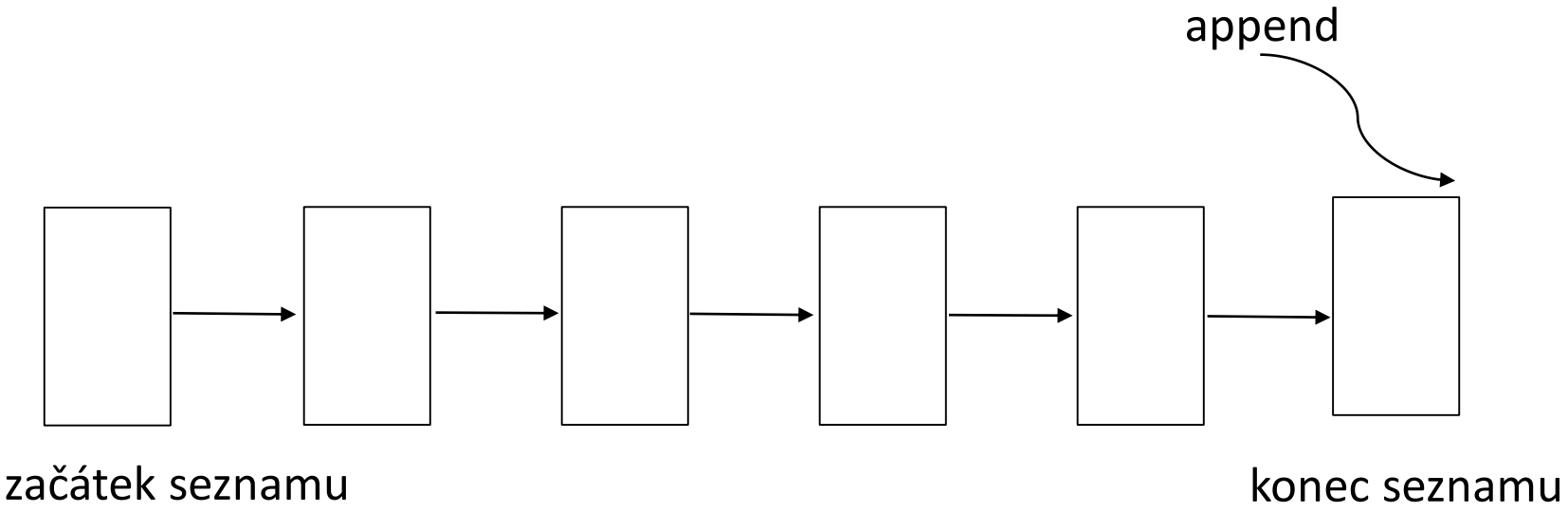
Operace kontejneru:

- Vytvořit prázdný kontejner (konstruktor, init)
- Zjistit počet uložených prvků (size)
- Oveřit prázdnost kontejneru (empty)
- Přístup k prvkům kontejneru (read, top, front, ...)
- Vložit prvek do kontejneru (insert, push, add, ...)
- Odstranit prvek z kontejneru (delete, pop, remove, ...)
- Vymazat všechny uložené prvky (clear)

Abstraktní datové typy

- Seznam – List
- Fronta – Query
- Zásobník – Stack
- Strom – Tree
- Tabulka – Table, Map
- Množina – Set

ADT – Seznam



ADT – Seznam

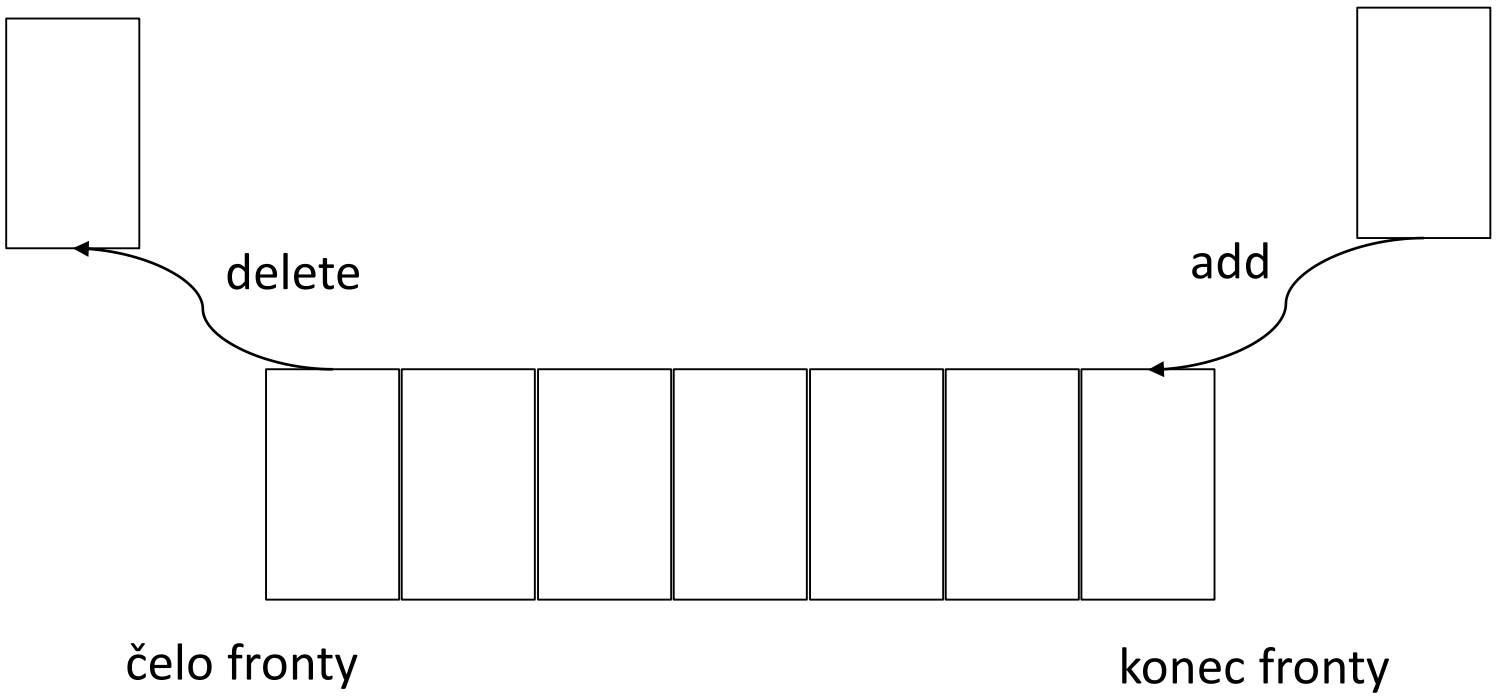
- Seznam je tvořen uspořádanou posloupností prvků (podobně jako pole).
- Každý prvek odkazuje na následníka nebo předchůdce.
- Uspořádání prvků na základě klíče.
- Průchod seznamem – sekvenční (na základě uspořádání).
- Využívá se rekurzivity seznamu – každý podseznam je také seznamem.

ADT – Seznam

Operace pro seznam:

- append(Y) – připojit prvek Y do seznamu
- count(Y) – vrátit počet položek Y
- index(Y) – vrátit nejmenší index odpovídající položce Y
- reverse – obrátit položky v seznamu
- insert(i, Y) – vložit prvek Y do seznamu na index i
- pop(i) – vrátit hodnotu i-tého prvku a odebrat jej ze seznamu
- sort() – setřídit seznam

ADT – Fronta



ADT – Fronta

FIFO - First In, First Out

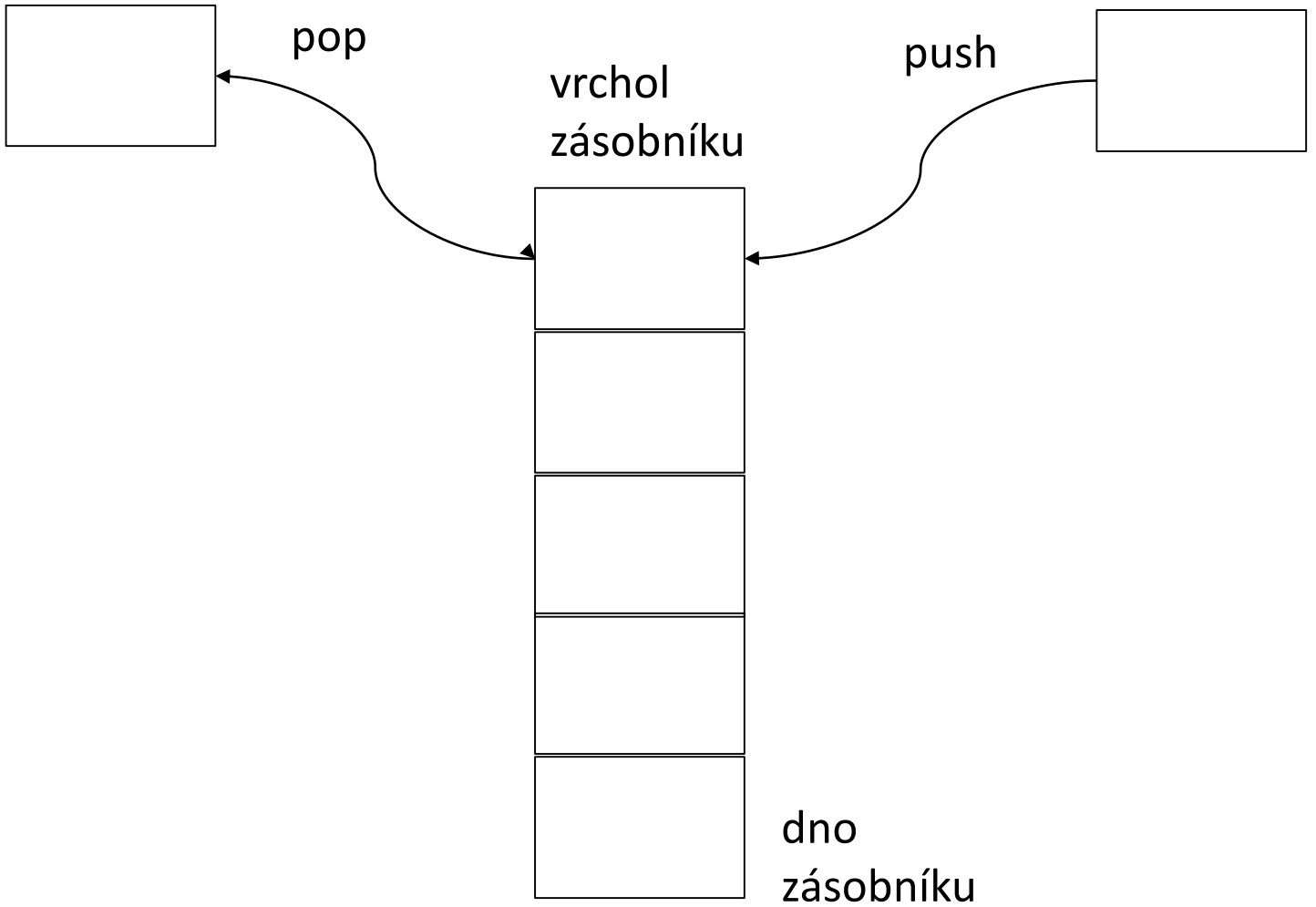
- Fronta se používá v případech, kdy potřebujeme zpracovávat (ukládat a vybírat) prvky v stejném pořadí.
- První uložený prvek bude jako první vybrán.
- Příklad:
  - Maily jsou řazeny do fronty a odesílány dle pořadí.
  - Speciální případ – fronta s prioritou. Je definována priorita prvků, prvky s vyšší prioritou mohou “předbíhat” prvky s nižší prioritou na výstupu (např. přenos packetů – Skype má přednost před maily)

ADT – Fronta

Operace pro frontu:

- add – vložit prvek do fronty
- delete – vybrat prvek z fronty (je odstraněn první prvek – hlava fronty)
- get – získat hodnotu prvního prvku fronty
- isEmpty – ověření, zda je fronta prázdná
- size – dotaz na počet prvků obsažených ve frontě

ADT – Zásobník



ADT – Zásobník

LIFO - Last In, First Out

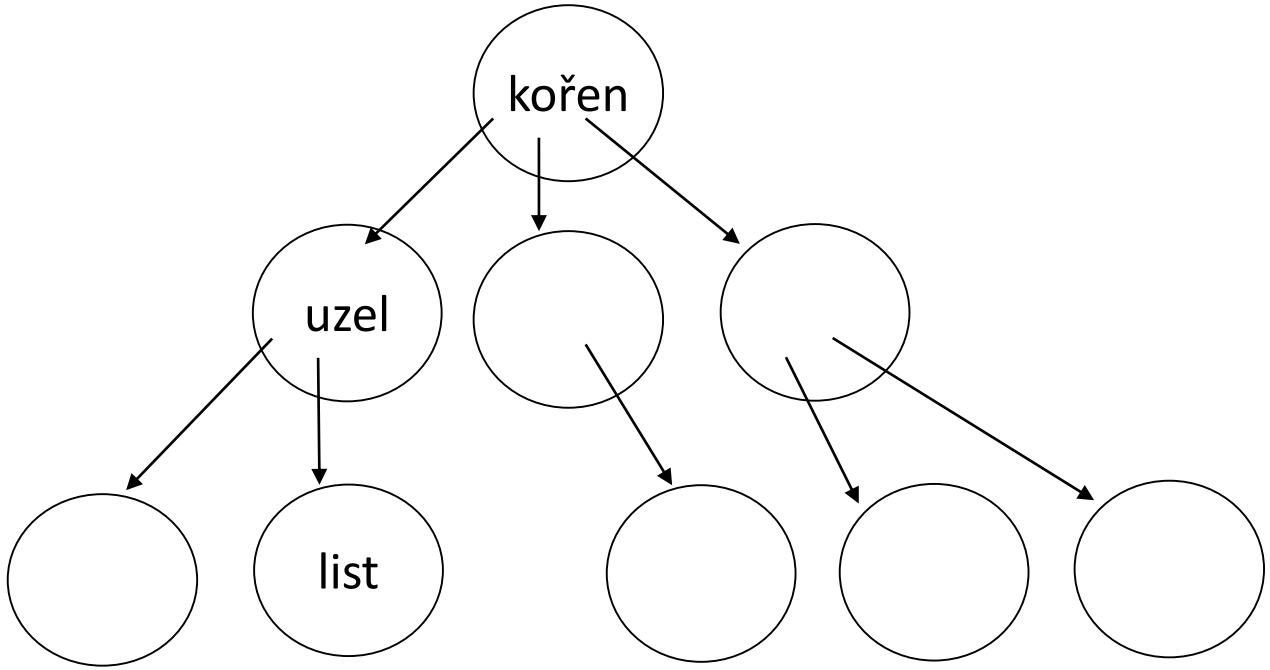
- Zásobník se většinou využívá v případech, kdy potřebujeme dočasně ukládat a vybírat prvky během jejich zpracování.
- První uložený prvek bude vybrán jako poslední – dno zásobníku.
- Posledně uložený prvek lze odebrat jako první – vrchol zásobníku.
- Příklad:
  - Rekurze, vyhodnocování výrazů.

ADT – Zásobník

Operace pro zásobník:

- push - vloží prvek na vrchol zásobníku
- pop - odstraní prvek z vrcholu zásobníku
- top - dotaz na hodnotu prvku z vrcholu zásobníku
- isEmpty – ověření, zda je zásobník prázdný

ADT – Strom



ADT – Strom

- Strom lze chápat jako zobecněný seznam. Každý prvek může mít více následníků.
- Prvky – uzly stromu, kořen stromu, listy. Hierarchická struktura.
- Vizualizace - kořen nahoře.
- Teorie grafů – každý souvislý graf bez kružnic.
- Rekurze – každý podstrom stromu je také stromem.
- Příklad: Vyhledávací strom, rozhodovací problémy.

ADT – Strom

Operace se stromem:

- vložení prvku na určitou pozici ve stromu
- vyhledání prvku
- vymazání prvku
- zjištění hloubky stromu, zjištění počtu prvků
- procházení stromem (do hloubky/do šířky)

Použití datových struktur

Základní úlohy

- třídění,
- vyhledávání,
- indexace.

DT definované uživatelem

- uživatel při řešení může potřebovat i jiné datové typy, než nabízí prostředí programovacího jazyka.
- většina jazyků proto umožňuje programátorovi definovat vlastní datové typy.
- mluvíme pak o uživatelem definovaných datových typech (odvozených DT).
- zavádí se tehdy, pokud více proměnných má logickou návaznost a jako celek reprezentují nějaký objekt
  - *např. auto (délka, počet sedadel, objem motoru, cena)*
- tvorbu vlastních DT nám umožňují struktury

Děkuji za pozornost

Michal Kačmařík

michal.kacmarik@vsb.cz

www.vsb.cz