IB113 Úvod do programování a algoritmizace

Cvičení 10

Jaromír Plhák

Kódy z minula (množiny, slovník)

Datový typ

- Datový typ = množina hodnot s předdefinovanými operacemi na těchto hodnotách
  - Příklad: bool = ({True, False}; and, or, not)

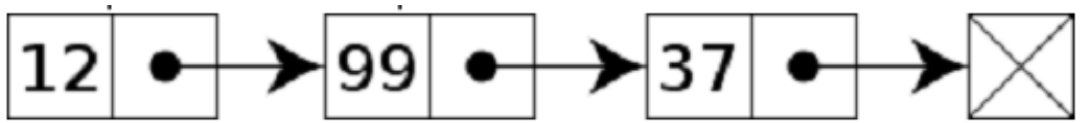
Čísla	Posloupnosti	
int	str	set
bool	list	dict
float	tuple	None
complex		

Abstraktní datový typ (ADT) – intuitivně

- Abstraktní datový typ = popis nějakých „dat“ + soupis **hlaviček funkcí**, které se váží k těmto „datům“
  - Slovo „data“ je použité velmi volně: může znamenat soubor, výtah, zvíře, nebo prakticky libovolný objekt
  - Dané funkce popisují programu, jak pracovat s těmito „daty“ a jak je měnit
  - Konkrétní implementace závisí na programátorovi
- ADT soubor: open(), read(), write(), close()
- ADT výtah: check_weight(), move_to_floor(), report_current_floor()
- ADT zvíře: eat(), drink(), move(), make_noise(), reproduce(), sleep()

Abstraktní datový typ (ADT) – programátorsky

- Abstraktní datový typ = rozhraní datové struktury
  - Implementačně nezávislá, matematická specifikace datové struktury s povolenými **operacemi** nad ní
  - Příklad: Seznam je posloupnost prvků, nad kterou můžeme vykonávat tyto (abstraktní) operace:
    - create() – vytvoření prázdného seznamu
    - add(x) – přidání prvku x do seznamu
    - remove(x) – odebrání prvku x ze seznamu
    - contains(x) – kontrola zda je prvek x v seznamu
    - is_empty() – kontrola na prázdnost seznamu
    - length() – počet prvků v seznamu
  - Konkrétní chovaní (a případně i názvy) těchto operací závisí od programovacího jazyka
  - Jedna z implementací: DS list v Pythonu
  - Jiná implementace: DS LinkedList v Javě:



Zásobník – vlastnosti

- Speciální typ seznamu
  - Přidávat a odebírat prvky můžeme jen na jednom jeho konci (tzv. *vrchol zásobníku)*
- Nesmíme přidávat ani odebírat prvky na druhém jeho konci (tzv. *dno zásobníku*) ani nikde uprostřed
- LIFO (Last In – First Out)

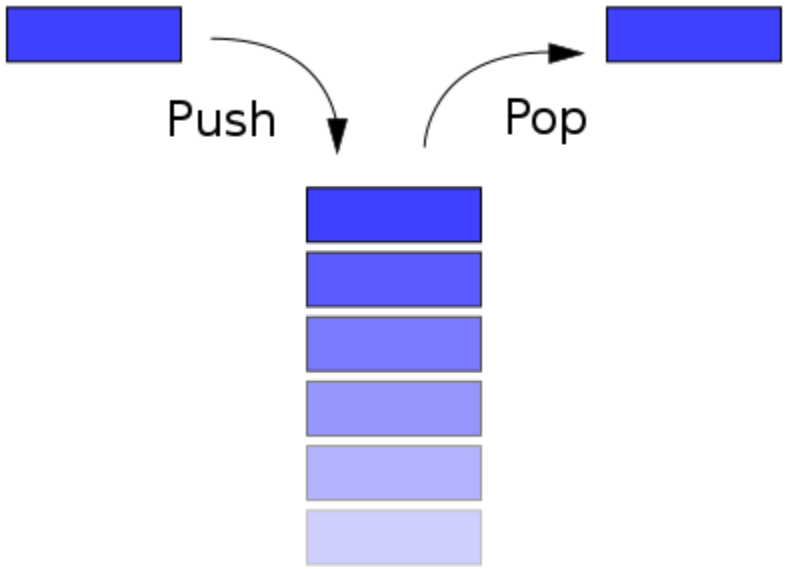
Zdroj: http://galleryhip.com/stack-of-plates.html

Zdroj: http://gebjgbtl231.webnode.cz/products/recenze-zasobnik-h-kg36-50-ran-src-/

Zdroj: http://www.billfrymire.com/gallery/balance-stability-rocks.jpg.html

Zásobník - vlastnosti

- Zásobník podporuje tyto (abstraktní) operace:
  - init() – vytvoření prázdného zásobníku
  - is_empty() – kontrola na prázdnost zásobníku
  - push(x) – přidání prvku x na vrchol zásobníku
  - pop() – odebrání prvku z vrcholu zásobníku
  - top() – náhled na prvek na vrcholu zásobníku



Zdroj: http://en.wikipedia.org/wiki/File:Data_stack.svg

Zásobník – simulace operací v Pythonu

- init() – vytvoření prázdného zásobníku

```
>>> stack = []
```

- push(x) – přidání prvku x na vrchol zásobníku

```
>>> stack.append (1)
>>> stack.append (2)
>>> stack.append (3)
>>> stack
[1, 2, 3]
```

- pop() – odebrání prvku z vrcholu zásobníku

```
>>> stack.pop ()
3
>>> stack
[1, 2]
```

Úkol

- 9.5.3. Kontrola uzávorkování
  - Napište funkci, která pro zadaný řetězec složený pouze ze závorek [](){} ověří, že jde o korektní uzávorkování.

```
def parenthesis_check(value):
        pass


print(parenthesis_check('([]({()}))[]{[()]}')) # True
print(parenthesis_check('([)]')) # False
```

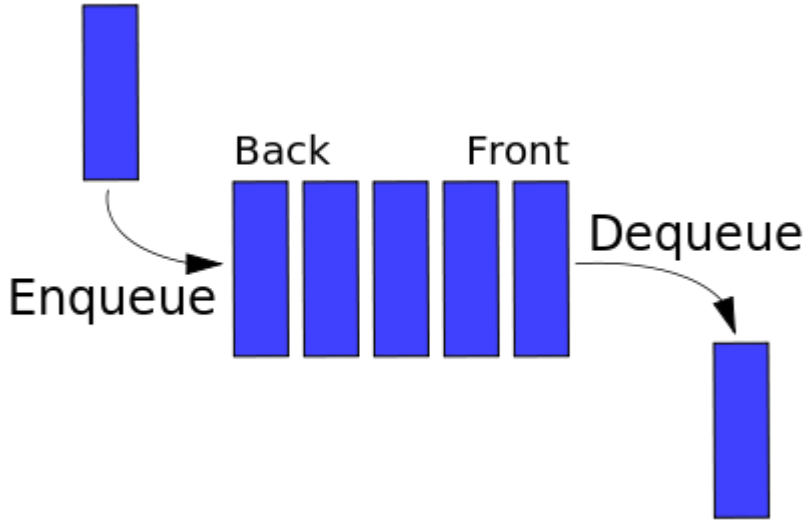
Fronta

- Speciální typ seznamu
- Přidávat prvky smíme jen na jednom konci (tzv. *začátek fronty)*
- Odebírat prvky smíme jen na druhém konci (tzv. *konec fronty)*
- FIFO (First In – First Out)

Zdroj: http://www.mathworks.com/matlabcentral/fileexchange/24238-priority-queue--mex-c++-

Fronta - vlastnosti

- Fronta podporuje tyto (abstraktní) operace:
  - init() – vytvoření prázdné fronty
  - is_empty() – kontrola na prázdnost fronty
  - enqueue(x) – přidání prvku x na konec fronty
  - dequeue() – odstránení prvku, který „je právě na řadě“
  - first() – náhled na první prvek



Zdroj: http://en.wikipedia.org/wiki/File:Data_Queue.svg

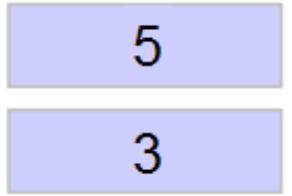
Úkol

- 9.2.2. Horký brambor
  - Horký brambor je jednoduchá dětská hra, kterou hraje n lidí s jedním horkým bramborem, který si mezi sebou přehazují. Kdo má brambor vyhodnotí jednoduché kritérium (jako každý sedmý) a pokud toto kritérium splňuje, pak vypadává ze hry. Vyhrává poslední hráč.
  - Naprogramujte tuto hru **za použití fronty**. Přičemž hraje n lidí, jejichž jména dostane program v seznamu a vypadává každý k-tý hráč. Průběh hry, vítěze a počet kol průběžně vypisujte.

Zásobník – reverzní polská notace

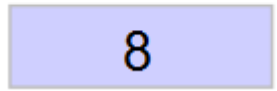
- Zadání: řetězec 3 5 + 7 2 - *
- Čteme zleva doprava:
  - Pokud přečteme číslo, uložíme ho na zásobník
  - Pokud přečteme operátor, vybereme poslední dvě čísla, aplikujeme na ně daný operátor a výsledek uložíme na zásobník
- **1** Přečti 3, ulož 3 na zásobník, zůstává 5 + 7 2 - *

- **2** Přečti 5, ulož 5 na zásobník, zůstává + 7 2 - *

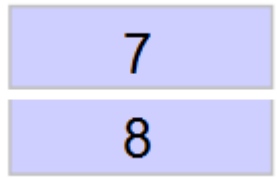


Zásobník – reverzní polská notace

- **3** Přečti +, vyber 5 ze zásobníku, vyber 3 ze zásobníku, spočti 3 + 5, ulož 8 na zásobník, zůstává 7 2 - *



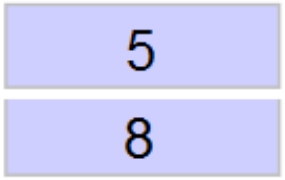
- **4** Přečti 7, ulož 7 na zásobník, zůstává 2 - *



- **5** Přečti 2, ulož 2 na zásobník, zůstává - *

Zásobník – reverzní polská notace

- **6** Přečti -, vyber 2 ze zásobníku, vyber 7 ze zásobníku, spočti 7 - 2, ulož 5 na zásobník, zůstává *



- **7** Přečti *, vyber 5 ze zásobníku, vyber 8 ze zásobníku, spočti 8 * 5, ulož 40 na zásobník, konec

Úkol

- 9.1.1. Interpretace výrazu v postfixu
  - Napište funkci, která obdrží řetězec v postfixové notaci a ta výraz za pomocí zásobníku vyhodnotí. Implementuje operace +, -, / a * v plovoucí čárce.
  - Tip: může se vám hodit metoda split, která je použitelná na řetězcích: "a b c".split().