Datové typy, práce s daty

IB111 Programování a algoritmizace

2011

Data a algoritmizace

- jaká data potřebuji pro vyřešení problému?
- jak budu data reprezentovat?
- jaké operaci s nimi potřebuji provádět?
- dobré myšlení o datech je klíčové

Abstraktní datový typ (ADT)

- specifikuje operace a jejich funkcionalitu (příp. složitost)
- nezávisí na konkrétní implementaci
- nyní: přehled základních ADT
- později: některé klasické implementace

Důležité principy: abstrakce, oddělení specifikace a implementace

Zásobník (stack)

- operace:
  - push – vložení
  - pop – odstranění
  - top – horní prvek
  - empty – test prázdnosti
- LIFO = Last In First Out
- intuitivní příklad: sloupec knih
- použití: procházení grafů, vyhodnocování výrazů, rekurze

Python: zásobník pomocí seznamu

```
>>> stack = [3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack
[3, 4, 5, 6]
```

Fronta (queue)

- operace:
  - push – vložení
  - pop (shift) – odstranění
  - top (first) – první prvek
  - empty – test prázdnosti
- FIFO = First In First Out
- příklad: zpracování příchozích požadavků serverem

Python a fronta

- lze pomocí seznamu, ale pomalé
- použití knihovny `collections`

Množina (set)

- operace:
  - insert – vložení
  - find – test na přítomnost
  - delete – odstranění
- příklady:
  - evidování navštívených vrcholů při prohledávání
  - počítání „průniku“ informací ze dvou souborů

Slovník (dictionary, map)

- zobecnění množiny, pole
- dvojice klíč, hodnota
  - klíče jsou unikátní
  - neuspořádané
- příklady:
  - počet výskytů jednotlivých slov v textu
  - „kešování“ výsledků časově náročných výpočtů

Python: dictionary

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
```

Praktický příklad: práce s daty

- dva soubory (tabulky) s botanickými daty
  - okolní vegetace
  - semínka ve vzorcích
- sloupce: lokality
- řádky: druhy rostlin (seznam druhů se liší v souborech)
- úkol: zkombinovat do jednoho souboru pro další analýzy
- využití datového typu slovník

Metodika práce s daty

Oddělení dat a funkcionality

Důležitý princip v mnoha oblastech informatiky

Příklad web

- webový portál: oddělení funkcionality, dat a vzhledu
- typická realizace:
  - funkcionalita – program (PHP)
  - data – databáze (MySQL)
  - vzhled – grafický styl (CSS)

Konkrétní ilustrace: převodník kódování

- převod textu na různá kódování
- morseova abeceda, braillovo písmo, ...
- jak vypadá program bez oddělení dat od funkcionality?
- jak vypadá program s odděleními daty?

Převodník: nevhodné řešení

```
def prevod_morse(s):
    vystup = ''
    for i in range(len(s)):
        if s[i] == 'A': vystup += '.-|'
        elif s[i] == 'B': vystup += '-...|'
        elif s[i] == 'C': vystup += '-.-.|'
        elif s[i] == 'D': vystup += '-..|'
        # atd
    return vystup
```

Převodník: řešení se slovníkem

```
morse = {'A':'.-', 'B': '-...', 'C': '-.-.',
         'D': '-..' } # atd

def prevod_morse(s):
    vystup = ''
    for i in range(len(s)):
        if morse.has_key(s[i]):
            vystup += morse[s[i]] + '|'
    return vystup
```

Převodník: řešení s polem

```
morse = ['.-', '-...',  '-.-.', '-..'  ] # atd

def prevod_morse(s):
    vystup = ''
    for i in range(len(s)):
        if ord('A') <= ord(s[i]) <= ord('Z'):
            c = ord(s[i]) - ord('A')
            vystup += morse[c] + '|'
    return vystup
```

Převodník: stále to není ideální

```
def prevod(s, kodovani):
    if kodovani == 'morse':
        return prevod_morse(s)
    elif kodovani == 'braille':
        return prevod_braille(s)
    elif kodovani == 'semafor':
        return prevod_semafor(s)
    # atd
```

Převodník: další krok

```
sub = {
    "morse": [".-","-...","-.-.","-..", # atd ],
    "braille": ["BWWWWW","BWBWWW","BBWWWW", # atd ],
}

def prevod(s, kodovani):
    vystup = ''
    if not sub.has_key(kodovani): return None
    for i in range(len(s)):
        if ord('A') <= ord(s[i]) <= ord('Z'):
            c = ord(s[i])-ord('A')
            vystup += sub[kodovani][c] + '|'
    return vystup
```

Shrnutí

- abstraktní datový typ: koncept
- zásobník, fronta, množina, slovník
- princip oddělení dat a funkcionality