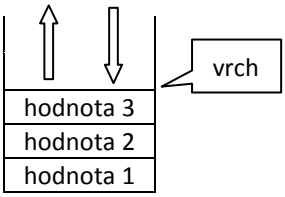
**Zásobník**




je abstraktný dátový typ, na ktorom sú dovolené len operácie:

- vytvoriť prázdny zásobník
- pridať prvok na vrch zásobníka
- odobrať prvok z vrchu zásobníka
- zistiť, či je zásobník prázdny

Nie je možné (dovolené) prechádzať zásobníkom!

Zásobník nám umožňuje dočasne uložiť údaje a spracovať ich až neskôr - v opačnom poradí, ako boli vložené. Zásobník sa označuje aj ako štruktúra LIFO (Last In - First Out, posledný dnu – prvý von). Zásobník, ako typ pamäte, sa používa na odkladanie návratových adries pred odchodom do podprogramov, na odkladanie hodnôt lokálnych premenných, na odkladanie „vonkajších častí“ vyhodnocovaných aritmetických výrazov pri vnáraní do nich, pri metóde „rozdeľ a panuj“, pri prehľadávaní do hĺbky a pod.

Pomocou metód zoznamu append() a pop() možno simulovať operácie zásobníka.

```
>>> zas = []                  # vytvorenie prázdneho zásobníka
>>> zas                       # vypísanie obsahu zásobníka
[]
>>> zas.append(1)             # pridanie hodnoty 1 do zásobníka
>>> zas
[1]
>>> zas.append(2)             # pridatie hodnoty 2 do zásobníka
>>> zas
[1, 2]
>>> zas.pop()                 # odobratie hodnoty zo zásobníka
2
>>> zas
[1]
>>> zas.pop()                 # odobratie hodnoty zo zásobníka
1
>>> zas
[]
>>> zas.pop()                 # pokus o odobratie hodnoty z prázdneho zásobníka
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    zas.pop()
IndexError: pop from empty list
>>>
```

Príklad Z.1

Vytvorte program na simulovanie práce zásobníka. Program vytvorí prázdny zásobník a umožní vkladať hodnoty na vrch zásobníka, odoberať hodnoty z vrchu zásobníka a „pozrieť sa“, aká hodnota je na vrchu zásobníka.

Príklad ponuky:

```
             Z Á S O B N Í K
Vložiť na vrch ..................... 1
Odobrať z vrchu .................... 2
Hodnota na vrchu ................... 3
Koniec ............................. ?
Tvoja voľba?
```

Riešenie:

```
def ponuka():
    while True:
        print(12*" ","Z Á S O B N Í K")
        print("Vložiť na vrch ..................... 1")
        print("Odobrať z vrchu .................... 2")
        print("Hodnota na vrchu ................... 3")
        print("Koniec ............................. ?")
```

```
        odpoved = input("Tvoja voľba? ")
        if odpoved == "1":
            vlozit()
        elif odpoved == "2":
            vybrat()
        elif odpoved == "3":
            hodnotaNaVrchu()
        else:
            break

def vlozit():
    hodnota = input("Vložiť hodnotu: ")
    zas.append(hodnota)
    print()
    print("Vložil som hodnotu:", hodnota)
    print()

def vybrat():
    print()
    if len(zas) > 0:
        hodnota = zas.pop()
        print("Odobral som hodnotu:", hodnota)
    else:
        print("Zásobník je prázdny!")
    print()

def hodnotaNaVrchu():
    print()
    if len(zas) > 0:
        print("Hodnota na vrchu:", zas[-1])
    else:
        print("Zásobník je prázdny!")
    print()

# ================== HLAVNÝ PROGRAM ==================
zas = []          # po spustení programu sa vytvorí prázdny zásobník!
print("Vytvoril som prázdny zásobník!")
print()
ponuka()
```

Príklad Z.1 doplňte o funkciu na zistenie a vypísanie, či je zásobník prázdny.
Ozrejmite si vzťah medzi hodnotou len(zásobník) a vrch zásobníka.

Príklad Z.2
Funkcie z príkladu Z.1 zovšeobecnite použitím parametrov a predpokladajte, že zásobník má konečnú veľkosť - kapacitu. Ponuku doplňte voľbou: „Je zásobník prázdny?“.

```
def vlozit(zas):
    print()
    if len(zas) < KAPACITA_ZASOBNIKA:          # test, či nie je zásobník plný!
        hodnota = input("Vložiť hodnotu: ")
        zas.append(hodnota)
        print("Vložil som hodnotu:", hodnota)
    else:
        print("Zásobník je plný!")
    print()

def vybrat(zas):
    print()
    if len(zas) > 0:    # hodnota len(zas) napovedá o pozícii vrchu zásobníka
        hodnota = zas.pop()
        print("Odobral som hodnotu:", hodnota)
    else:
```

```
        print("Zásobník je prázdny!")
    print()

def hodnotaNaVrchu(zas):
    print()
    if len(zas) > 0:
        print("Hodnota na vrchu:", zas[-1])
    else:
        print("Zásobník je prázdny!")
    print()

def jePrazdny(zas):
    print()
    if len(zas) == 0:
        print("Zásobník je prázdny!")
    else:
        print("Zásobník nie je prázdny!")
    print()

# ================ HLAVNÝ PROGRAM ==================

KAPACITA_ZASOBNIKA = 5         # maximálne päť hodnôt
z1 = []
print("Vytvoril som prázdny zásobník!")
print()

while True:
    print(12*" ","Z Á S O B N Í K")
    print("Vložiť na vrch ..................... 1")
    print("Odobrať z vrchu .................... 2")
    print("Hodnota na vrchu ................... 3")
    print("Je prázdny? ........................ 4")
    print("Koniec ............................. ?")
    odpoved = input("Tvoja voľba? ")
    if odpoved == "1":
        vlozit(z1)
    elif odpoved == "2":
        vybrat(z1)
    elif odpoved == "3":
        hodnotaNaVrchu(z1)
    elif odpoved == "4":
        jePrazdny(z1)
    else:
        break
```

Príklad Z.3

Vytvorte program, ktorý umožní pomocou voľby vložiť hodnotu do zásobníka 1 alebo vložiť hodnotu do zásobníka 2 a na voľbu odobrať odoberie hodnotu z toho zásobníka, v ktorom je väčšia hodnota na vrchu. Ak sú v oboch zásobníkoch na vrchoch rovnaké hodnoty, odoberie obe. Ak je jeden zo zásobníkov prázdny, ďalej odoberá hodnoty z neprázdneho zásobníka, až kým nie sú oba zásobníky prázdne.
Pri riešení vytvorte a použite funkcie zlozit(zas), odobrat(zas), jePrazdny(zas) a hodnotaNaVrchu(zas).

Príklad ponuky:

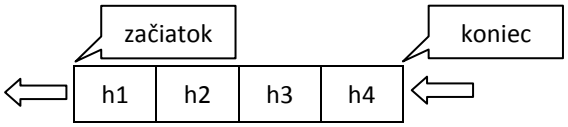
```
          Z Á S O B N Í K Y
Vložiť do zásobníka 1 ..................... 1
Vložiť do zásobníka 2 ..................... 2
Odobrať zo zásobníka s väčšou hodnotou .... 3
Koniec ..................................... ?
Tvoja voľba?
```

```
def vlozit(zas):
    print()
    hodnota = input("Vložiť hodnotu: ")
    zas.append(hodnota)
    print("\nVložil som hodnotu:", hodnota)
    print()

def odobrat(zas):
    hodnota = zas.pop()
    print("Odobral som hodnotu:", hodnota)

def jePrazdny(zas):
    return len(zas) == 0

def hodnotaNaVrchu(zas):
    return zas[-1]          # netestuje, či nie je zásobník prázdny!

def odobratVacsiuHodnotu(zas1, zas2):
    print()
    if jePrazdny(zas1) and jePrazdny(zas2):
        print("Zásobníky sú prázdne!")
    else:
        if jePrazdny(zas1):
            odobrat(zas2)
        elif jePrazdny(zas2):
            odobrat(zas1)
        else:
            if hodnotaNaVrchu(zas1) > hodnotaNaVrchu(zas2):
                odobrat(zas1)
            elif hodnotaNaVrchu(zas1) < hodnotaNaVrchu(zas2):
                odobrat(zas2)
            else:
                odobrat(zas1)
                odobrat(zas2)
    print()

# ================= HLAVNÝ PROGRAM ==================
z1 = []
z2 = []
print("Vytvoril som prázdne zásobníky z1 a z2!")
print()

while True:
    print(10*" ","Z Á S O B N Í K Y")
    print("Vložiť do zásobníka 1 ..................... 1")
    print("Vložiť do zásobníka 2 ..................... 2")
    print("Odobrať zo zásobníka s väčšou hodnotou .... 3")
    print("Koniec ..................................... ?")
    odpoved = input("Tvoja voľba? ")
    if odpoved == "1":
        vlozit(z1)
    elif odpoved == "2":
        vlozit(z2)
    elif odpoved == "3":
        odobratVacsiuHodnotu(z1,z2)
    else:
        break
```

**Rad**

je abstraktný dátový typ, na ktorom sú dovolené len operácie:

- vytvoriť prázdny rad
- pridať prvok na koniec radu
- odobrať prvok zo začiatku radu
- zistiť, či je rad prázdny




Rad nám umožňuje dočasne uložiť údaje a spracovať ich neskôr v poradí, ako boli vložené. Označuje sa aj ako štruktúra FIFO (First In - First Out, prvý dnu – prvý von). Rad, ako typ pamäte, sa používa na radenie úloh prichádzajúcich s požiadavkou na tlač (s rovnakou prioritou), pri prehľadávaní do šírky a pod. Český názov pre rad je fronta.

Príklad R.1
Vytvorte program na simulovanie práce radu. Program vytvorí prázdny rad a umožní vkladať hodnoty na koniec radu, odoberať hodnoty zo začiatku radu a zistiť, či je rad prázdny.

```
def ponuka():
    while True:
        print(15*" ","R A D")
        print("Vložiť na koniec ..................... 1")
        print("Odobrať zo začiatku .................. 2")
        print("Hodnota na začiatku .................. 3")
        print("Je prázdny? .......................... 4")
        print("Koniec ............................... ?")
        odpoved = input("Tvoja voľba? ")

        if odpoved == "1":
            vlozit()
        elif odpoved == "2":
            vybrat()
        elif odpoved == "3":
            hodnotaNaZaciatku()
        elif odpoved == "4":
            jePrazdny()
        else:
            break

def vlozit():
    hodnota = input("Vložiť hodnotu: ")
    rad.append(hodnota)
    print()
    print("Vložil som hodnotu:", hodnota)
    print()

def vybrat():
    print()
    if len(rad) > 0:
        hodnota = rad.pop(0)
        print("Odobral som hodnotu:", hodnota)
    else:
        print("Rad je prázdny!")
    print()

def hodnotaNaZaciatku():
    print()
    if len(rad) > 0:
        print("Hodnota na zaciatku:", rad[0])
    else:
        print("Rad je prázdny!")
    print()
```

```
def jePrazdny():
    print()
    if len(rad) == 0:
        print("JE prázdny!")
    else:
        print("NIE JE prázdny!")
    print()

# ================ HLAVNÝ PROGRAM =================
rad = []
print("Vytvoril som prázdny rad!")
print()
ponuka()
```

Príklad R.2

Použitie funkcií z príkladu R.1 zovšeobecnite doplnením o parametre a použite v situácii, keď sú dané dva rady a prednostne sa odoberá

a) z radu 1, a až keď je prázdny, z radu 2
b) z radu, v ktorom je na začiatku väčšia (menšia) hodnota

Príklad R.3

Zákazník príde do čakárne a „klikne na tlačidlo“ Príchod. Vytvorte program, ktorý umožní

1. evidovať čas príchodu zákazníka do čakárne,
2. odstránenie z radu po voľbe Odchod a vypísanie času stráveného v zariadení,
3. vypísanie počtu čakajúcich zákazníkov.

Ukážka:

```
Príchod ......................... 1
Odchod .......................... 2
Počet čakajúcich ................ 0
Koniec .......................... ?
Vlož voľbu: 1
Príchod 11:24:33

Príchod ......................... 1
Odchod .......................... 2
Počet čakajúcich ................ 0
Koniec .......................... ?
Vlož voľbu: 2
Odchod 11:24:49
Dĺžka obsluhy 0:00:15
```

Využite funkciu datetime.now() modulu datetime (pozri študijný text Použitie štandardných funkcií na prácu s reťazcami - úloha 7).

```
import datetime

def vloz():
    rad.append(datetime.datetime.now())
    print("Príchod", str(rad[-1])[11:19])
    print()

def odober():
    if len(rad) > 0:
        prisiel = rad.pop(0)
        odisiel = datetime.datetime.now()
        print("Odchod", str(odisiel)[11:19])
        print("Dĺžka obsluhy", str(odisiel-prisiel)[:7])
        print()
    else:
        print("Musíš najprv zaznamenať príchod!\n")
```

```
def pocetCakajucich():
    print("Počet čakajúcich:", len(rad))
    print()


# ====================== HLAVNÝ PROGRAM ======================
rad = []
while True:
    print("Príchod ....................... 1")
    print("Odchod ........................ 2")
    print("Počet čakajúcich ............... 0")
    print("Koniec ........................ ?")
    odpoved = input("Vlož voľbu: ")

    if odpoved == "1":
        vloz()
    elif odpoved == "2":
        odober()
    elif odpoved == "0":
        pocetCakajucich()
    else:
        break
```

Príklad R.3 doplňte o poznatok: Obsluha zariadenia je schopná denne obslúžiť najviac 5 zákazníkov.

Príklad R.4

Klienti prichádzajú k výťahu a stavajú sa do radu (nech program po voľbe Prišiel klient vygeneruje náhodné celé číslo od 15 do 130 kg a zaradí ho do radu čakajúcich). Po príchode výťahu (voľba Prišiel výťah) nastupujú klienti do výťahu. Vo výťahu je tlaková váha, ktorá automaticky určí súčet hmotností ľudí vo výťahu. Ak súčet ich hmotností prekročí nosnosť výťahu, klient, ktorý nastúpil posledný, musí vystúpiť. Vytvorte program simulujúci opísaný dej.

```
import random

def ponuka():
    while True:
        print("Prišiel klient ............... 1")
        print("Prišiel výťah ................ 0")
        print("Koniec ....................... ?")
        odpoved = input("Nastala udalosť: ")
        if odpoved == "1":
            prisielKlient()
        elif odpoved == "0":
            prisielVytah()
        else:
            break

def prisielKlient():
    hmotnost_klienta = random.randint(15,130)
    # hmotnosť klienta je náhodné celé číslo od 15 do 130 kg
    rad.append(hmotnost_klienta)
    print("Hmotnosť {} kg\n".format(rad[-1]))
```

```
def prisielVytah():
    if len(rad) == 0:
        print("Nik nečaká!\n")
    else:
        sucet_hmotnosti = rad[0]            # nastúpil do výťahu na test - z radu neodstráni!
        nastupia = 0
        while sucet_hmotnosti <= NOSNOST_VYTAHU:
            rad.pop(0)                      # môže zostať vo výťahu:) - až teraz odstráni z radu!
            nastupia += 1                   # ďalší sa môže odviezť výťahom
            if len(rad) > 0:                # ak ešte niekto čaká
                sucet_hmotnosti += rad[0]   # nastúpi do výťahu na test súčtu hmotností
            else:
                break
        print("Nastúpia: {}\n".format(nastupia))

# ================= HLAVNÝ PROGRAM ===============
NOSNOST_VYTAHU = 150 #kg
rad = []
ponuka()
```

Program R.4 doplňte o situáciu: ak je v rade viac ako 10 osôb, nech program oznámi: Použite schodište.
Program R.4 doplňte o hlásenie: ak je hmotnosť klienta nad 100 kg, nech program vypíše: Odporúčam použiť schodište!

Poznámka
Modelovanie odobratia prvku zo začiatku radu pri použití pop() typu list nie je časovo efektívne, pretože sa musia všetky prvky zoznamu posunúť (preindexovať) o jednu pozíciu doľava. Preto knižnica collections obsahuje špecializovaný typ deque s „rýchlymi“ funkciami append() a popleft().
Ukážka:

```
>>> from collections import deque
>>> rad = deque([])
>>> print(rad)
deque([])
>>> rad.append("Peter")
>>> rad
deque(['Peter'])
>>> rad = deque(['Peter', 'Roman', 'Filip'])
>>> rad
deque(['Peter', 'Roman', 'Filip'])
>>> rad.append("Karol")
>>> rad
deque(['Peter', 'Roman', 'Filip', 'Karol'])
>>> print(rad.popleft())
Peter
>>> rad
deque(['Roman', 'Filip', 'Karol'])
>>> meno = rad.popleft()
>>> meno
'Roman'
>>> rad
deque(['Filip', 'Karol'])
```