APSTRAKTNE STRUKTURE PODATAKA

Stog i red

APSTRAKTNE STRUKTURE PODATAKA

- nove strukture podataka za rješavanje nekih problema
- stogovi, redovi i stabla
- definirane osnovne operacije
- ne moramo znati detalje vezane za pohranjivanje podataka i izvođenja zadanih operacija
- apstraktne strukture podataka (engl. Abstract Data Type – ADT).
- implementirati ćemo ih kao klase

MODUL COLLECTIONS – KLASA DEQUE

- iz modula collections koristit ćemo klasu deque – za nove klase
- slično kao što i imamo tipove podataka kao što su list, str...

- dvostrani red (engl. double-ended queue–deque, što se čita kao: "dek
  - možemo lako dodavati ili uzimati elemente s jednog ili drugog kraja
  - iz liste možemo lako dodati ili obrisati elemente s kraja liste s metodama pop() i append(), ali s bilokoje grugogo mjeste u listi je puno teže obrisat ili dodat elemente
  - koristit ćemo je za kreiranje svih ostalih apstraktnih struktura

METODE KLASE DEQUE

naziv metode	opis
`d.append(x)`	stavlja element `x` na desni kraj objekta `d`
`d.appendleft(x)`	stavlja element `x` na lijevi kraj objekta `d`
`d.clear()`	briše sve elemente objekta `d`
`d.extend(li)`	dodaje sve elemente liste `li` na desni kraj elementa `d`
`d.extendleft(li)`	dodaje sve elemente liste `li` na lijevi kraj objekta `d` (elementi će se pojaviti u `d` u obrnutom poretku)
`d.pop()`	skida i vraća element s desnog kraja objekta `d`, javlja pogrešku ako je `d` prazan
`d.popleft()`	skida i vraća element s lijevog kraja objekta `d`, javlja pogrešku ako je `d` prazan

PRIMJER KORIŠTENJA METODA

```
from collections import deque

d = deque()
print(d)
d.append(1)
d.append(2)
print(d)
d.appendleft("A")
d.append("B")
print(d)
d.pop()
d.popleft()
print(d)
d.extendleft([1, 2, "A"])
print(d)
print(len(d))
print(d[0], d[3])
```

STOG (ENGL. STACK)

- kao tanjuri koji stoje jedan na drugom ili knjige koje stoje na hrpi…
- LIFO struktura (engl. Last In First Out) - objekt koji je dodan posljednji, prvi će se skinuti
- klasa Stack – na kojoj ćemo imati sljedeće metode

naziv metode	opis
`s.push(t)`	stavlja element `t` na vrh stoga `s`
`s.pop()`	skida i vraća element s vrha stoga `s`
`s.isEmpty()`	vraća `True` ako je stog `s` prazan

OBJASNITE ŠTO ĆE BITI SPREMLJENO NA STOGU AKO PROVEDEMO SLJEDEĆI NIZ NAREDBI:

```
s = Stack()
s.push('A')
s.push('D')
s.push('B')
s.push('C')
s.push('E')
s.pop()
s.pop()
print(s)
s.isEmpty()
s.pop()
print(s)
```

OBJASNITE ŠTO ĆE BITI SPREMLJENO NA STOGU AKO PROVEDEMO SLJEDEĆI NIZ NAREDBI:

```
s = Stack()
s.isEmpty()
s.push(5)
s.push(1)
s.pop()
s.isEmpty()
s.push(3)
s.push(7)
print(s)
s.push(1)
s.push(4)
s.pop()
s.pop()
s.pop()
print(s)
```

DEFINIRANJE KLASE STACK

```
from collections import deque

class Stack(deque):
    def __init__(self):
        super().__init__()

    def push(self, v):
        self.append(v)

    def pop(self):
        if not self.isEmpty():
            return super().pop()
        return None

    def isEmpty(self):
        return len(self) == 0

def main():
    s = Stack()
    print(s)
    print(s.isEmpty())
    s.push(2)
    s.push(4)
    print(s)
    print(s.pop())
    print(s)
main()
```

PRIMJER 1. (PISANJE ARITMETIČKIH IZRAZA)

- aritmetičke izraze najčešće na način da binarni operator pišemo između dvaju operanada (2 + 3) - infix-notacija
- aritmetički izraz možemo zapisivati i da prvo navedemo operande, a onda operator (2 3 +) - postfix-notacija
- ili da prvo navodimo operator, a onda operande (+ 2 3) - prefix-notacija
- kod prefiksnog i postfiksnog načina nije potrebno pisati zagrade

PRIMJER 1. (PISANJE ARITMETIČKIH IZRAZA)

1. Sljedeće aritmetičke izraze zapisane u infiksnim načinom zapišimo u prefix-notaciji i postfix-notaciji:

a) ((5 – 2) * 7) / 3

b) (2 + 3) * 4 – (8 + 2) / 5

c) ((7 – 8) / 4)*2 +(1 + 8) * 3

2. Sljedeće aritmetičke izraze zapisane u postfix-notaciji zapišimo u infix-notaciji

a) 5 2 + 4 - 3 *

b) 4 7 * 2 - 3 / 10 4 / -

3. Sljedeće aritmetičke izraze zapisane u prefix- notaciji zapišimo u infix-notaciji.

a) / * + 1 3 4 9

b) / * - 8 4 - 7 5 10

PRIMJER 1. PROGRAM

Napišimo program koji će izračunavati vrijednost aritmetičkog izraza zapisanog postfiksnim načinom, a koji se sastoji samo od osnovnih matematičkih operatora (+, -, * i /). Pretpostavka je da su operandi i operatori međusobno odvojeni znakom razmaka te su smješteni u stogu.

7 35 1 + 4 / -

RJEŠENJE PRIMJERA 1.

```
from stack import*
izraz = input().split()
print(izraz)
s = Stack()
for t in izraz:
    if t in "+-*/":
        a = s.pop()
        b = s.pop()
        if t == "+":
            s.push(b+a)
        elif t == "-":
            s.push(b-a)
        elif t == "*":
            s.push(b*a)
        else:
            s.push(b/a)
    else:
        s.push(int(t))
print(s.pop())
```

ZADATAK 1.

Napišimo program koji će izračunavati vrijednost aritmetičkog izraza zapisanog prefiksnom načinom, a koji se sastoji samo od osnovnih matematičkih operatora (+, -, * i /). Pretpostavka je da su operandi i operatori međusobno odvojeni znakom razmaka te su smješteni u stogu.

RED (ENGL. QUEUE)

- skupina ljudi koji čekaju u prodavaonici na blagajni
- FIFO struktura (engl. First in First out) - onaj koji je prvi ušao u red, prvi će i izaći iz njega

naziv metode	opis
`r.enqueue(t)`	stavlja element `t` na kraj reda `r`
`r.dequeue()`	skida i vraća prvi element reda `r`
`r.isEmpty()`	vraća `True` ako je red `r` prazan

OBJASNITE ŠTO ĆE BITI SPREMLJENO NA REDU AKO PROVEDEMO SLJEDEĆI NIZ NAREDBI:

```
r = Queue()
r.enqueue("X")
r.enqueue("Z")
r.enqueue("W")
r.enqueue("Y")
r.isEmpty()
r.dequeue()
r.dequeue()
r.enqueue("A")
r.dequeue()
```

OBJASNITE ŠTO ĆE BITI SPREMLJENO NA REDU AKO PROVEDEMO SLJEDEĆI NIZ NAREDBI:

```
r = Queue()
r.isEmpty()
r.enqueue(5)
r.enqueue(7)
r.enqueue(2)
r.enqueue(4)
r.enqueue(1)
r.dequeue()
r.dequeue()
r.dequeue()
r.isEmpty()
r.enqueue(6)
r.dequeue()
```

DEFINIRANJE KLASE RED

```
from collections import deque
class Queue(deque):
    def __init__(self):
        super().__init__()

    def enqueue(self, t):
        self.append(t)

    def dequeue(self):
        return self.popleft()

    def isEmpty(self):
        return len(self) == 0
```

```
def main():
    r = Queue()
    r.enqueue(2)
    r.enqueue(4)
    r.enqueue(3)
    r.enqueue(8)
    r.dequeue()
    r.dequeue()
    print(r)
    r.enqueue(1)
    r.enqueue(12)
    r.dequeue()
    r.dequeue()
    print(r.isEmpty())
    print(r)

main()
```

PRIMJER 1.

U školi se uskoro odvija Božićni koncert koji se odvija prema sljedećem rasporedu. Prvo je kratak „Uvod" u kojem voditelji koncerta započinju koncert i najavljuju pjesme koje će se u prvom dijelu pjevati. Nakon uvoda i pjesmi je kratka predstava „Božićna priča", a zatim opet voditelji najavljuju novi niz pjesama i na kraju voditelji završavaju koncert. Napišite program koji će točno navesti sve točke koje će biti na koncertu (uključujući i nastup voditelja i nazive pjesama) te osigurati da se po ispravnom redoslijedu izvršavaju.