8. Podatkovne strukture (sklad in vrsta)

Klasični pogled na programiranje daje prednost programom, ki delujejo nad podatki. Programi so strogo ločeni od podatkov. Običajno si podatke strukturiramo, tako, da ločimo abstraktno podatkovno strukturo od njene implementacije. Od implementacije zahtevamo, da natančno posnema abstraktne specifikacije, od uporabnika pa, da ne zlorabi morebitnih bližnjic, ki bi mu jih ponujale posamezne implementacije.

Podatkovno strukturo si lahko predstavljamo kot podatkovni tip skupaj z operacijami (metodami), ki jih lahko na njem izvajamo.

8.1. Sklad

Intuitivno si sklad predstavljamo kot kup, kot skladovnico, na katero nalagamo objekte in jih z vrha spet jemljemo.

8.2. Zgled: Denimo, da imamo niz, sestavljen iz znakov "(" in ")", v katerem je zapisan pravilni oklepajni izraz: npr. "","()","(())","()()", ... Napišite funkcijo, ki za dani niz ugotovi, ali gre za oklepajni izraz ali ne.

Ideja rešitve: imamo števec, ki pravi, koliko neuravnoteženih predklepajev imamo na danem koraku.

```
def oklepajniIzraz(niz):
      """Preverimo, ali je niz pravilen oklepajni izraz."""
      n = 0 # sklad je prazen
      for z in niz:
            if z == "(":
                  n += 1 # na sklad polozimo oznako
            elif z == ")":
                  n -= 1 # s sklada dvignemo oznako
            if n < 0:
                  return False # prevec zaklepajev
            else:
                  return False # niz ima druge znake
      if n == 0:
            return True # pravili oklepajni izraz
      else:
            return False # premalo zaklepajev
```

Tale primer je pokazal preprost sklad, kjer nam nikoli ni bilo potrebno pogledati na vrh sklada. Zgled lahko malo zakompliciramo.

8.3. Zgled: Denimo, da imamo niz, sestavljen iz znakov "(", ")", "[" in "]", v katerem je zapisan pravilni oklepajni izraz iz dveh vrst oklepajev: npr. "","()","[()]","[]()", ... Napišite funkcijo, ki za dani niz ugotovi, ali gre za oklepajni izraz ali ne. V tem primeru bi bilo dobro namesto števila n uporabljati niz, v katerega bi shranjevali predklepaje, zaklepaji pa bi jih iz niza odstranjevali.
Pri skladu elemente vstavljamo na istem koncu, s katerega jih odstranjujemo. Pri vrsti pa elemente vstavljamo na enem koncu (na repu), odstranjujemo pa jih na drugem koncu (na čelu).

Formalno lahko sklad opišemo tako, da določimo operacije s katerimi delujemo na sklad. Drugače ne smemo dostopati do sklada. Če se držimo takšnega dogovora je anš program neodvisen od dejanske implementacije sklada oz. teh operacij. Program nam ni potrebno spreminjati, tudi če zamenjamo implementacijo sklada oz. operacij nad njim.
Operacije na skladu:

- vrh (element na vrhu sklada)
- vstavi (element na vrh sklada)
- briši (element z vrha sklada)
- jeprazen (da ali ne)
- pripravi (prazen sklad)

Sklad lahko implementiramo z nizi npr. takole:

```
def vrh(sklad):
   """ vrne vrh sklada sklad."""
   return sklad[0]  # sklad mora biti neprazen!!!

def vstavi(element, sklad):
   """vrnemo nov sklad z elementom na vrhu."""
   return element+sklad

def brisi(sklad):
   """ vrnemo sklad skrajšan za element na vrhu."""
   return sklad[1:]

def jeprazen(sklad):
   """ je res, če je sklad prazen."""
   return len(sklad) == 0

def pripravi():
   """ vrne prazen sklad."""
   return ""

def izpisi(sklad):
   print("sklad = ",sklad)
```

Dodali smo še funkcijo, ki izpiše v vrstico vsebino sklad. Vrh je na levi strani.

8.4. Zgled: Denimo, da imamo niz nad petimi znaki "xf(),". Pri tem je vloga oklepajove očitna. "f" označuje eno- ali več-mestno funckijo, "x" spremenljivko, vejica pa loči argumente med seboj. Napišite funkcijo, ki za dani niz ugotovi, ali gre dobro oblikovan izraz ali ne.

```
def jeizraz(niz):
   """Izraz je oblike x ali pa je oblike f(a,b,...), kjer so a, b,
... izrazi.
Npr. x, f(x),f(x,x),f(f(x),x), so izrazi (x) ali f((f(x))) pa nista
izraz.
Dopustni znaki so:
x
f
(
)
,
"""
   # uporabljamo sklad. Na sklad zalagamo simbole:
   #  i ... pričakujemo izraz
   #  k ... končali smo z izrazom
   #  ( ... obdelujemo funkcijo
   sklad = pripravi()
   sklad = vstavi("i",sklad)
   for z in niz:
      izpisi(sklad)
      print("znak = ",z)
      if jeprazen(sklad):
         return False
      v = vrh(sklad)
      if v == "i":
         if z not in "fx":
            return False
         if z == "x":
            sklad = brisi(sklad)
```

```
            sklad = vstavi("k",sklad)
         else: # z mora biti f
            sklad = brisi(sklad)
            sklad = vstavi("(",sklad)
      elif v == "k":
         if z not in "),":
            return False
         sklad = brisi(sklad)
         if jeprazen(sklad):
            return False
         v = vrh(sklad)
         if z == ")" and v == "(":
            sklad = brisi(sklad)
            sklad = vstavi("k",sklad)
         else: # z mora biti ,
            if jeprazen(sklad):
               return False
            sklad = vstavi("i",sklad)
      elif v == "(":
         if z not in "(":
            return False
         sklad = vstavi("i",sklad)
   print("sklad na koncu je : ",sklad)
   if jeprazen(sklad):
      return False
   if vrh(sklad) == "k":
      return True
```

Pri skladu elemente vstavljamo na istem koncu, s katerega jih odstranjujemo. Pri vrsti pa elemente vstavljamo na enem koncu (na repu), odstranjujemo pa jih na drugem koncu (na čelu).

8.5. Naloge.
1. Napišite funkcijo, ki naravno število pretvori v osmiški sistem.
2. Napišite funkcijo, ki naravno število pretvori v šestnajstiški sistem. Števke so 0-9 in A-F.

8.6. Vrsta

Vrsta je podobna skladu, le da vstavljamo podatke na enem koncu (repu), jemljemo pa jih z vrste na drugem koncu, (na čelu).
Operacije na vrsti:

- čelo (element na čelu vrste)
- vstavi (element na rep vrste)
- briši (element s čela vrste)
- jeprazna (da ali ne)
- pripravi (prazno vrsto)

Operacije na vrsti, če jo realiziramo z nizom `vrsta`:

- čelo (element na čelu vrste) `vrsta[0]`
- vstavi (element na rep vrste) `vrsta = vrsta + znak`
- briši (element s čela vrste) `vrsta = vrsta[1:]`
- jeprazna (da ali ne) `len(vrsta) == 0`
- pripravi (prazno vrsto) `vrsta = “”`

8.7 Zgled: Izpišite veliko število n po pet znakov v vrsti.

```
n = 2**100
s = str(n)
m = len(s)
print(s)
vrsta = ""
for znak in s:
   vrsta = vrsta + znak
   if len(vrsta) > 4:
      print(vrsta)
      vrsta = ""
if len(vrsta) > 0:
   print(vrsta)
```

8.8. Naloge:

1. Napišite funkcijo izpisuj(niz,k), ki izpisuje dani niz po k znakov v vrstico. Vrstica je lahko krajša, če zahteva znak "\n" skok v novo vrsto.
2. Napišite funkcijo, blagajni(niz) ki simulira kupce pred dvema blagajnama. Pri tem naj bo niz sode dolžine. Za sodi indeks 2*i pomeni ord(niz[2*i]) število sekund, ki pretečejo do prihoda novega kupca. Novi kupec je znak z = niz[2*i +1]. Prvi kupec se postavi v prvo vrsto, drugi v drugo, tretji pa spet v prvo. Čas, ki ga kupec z znakom z prebije pred blagajno je enak ord(z). Ko kupca postavite v vrsto, zabeležite pri njem tudi čas prihoda. Ko pa je kupec postrežen, zapišite njegov indeks, čas, ki ga je prebil v sistemu ter številko blagajne, kjer je bil postrežen. Na koncu vrnite skupni čas delovanja sistema.
3. Ponovite drugo nalogo, le da se kupec vedno postavi v najkrajšo vrsto. (Če sta obe vrsti enako dolgi, se postavi v prvo vrsto.)
4. Ponovite 2. nalogo, le da je vrsta ena sama, kupec, ko pride na vrsto pa se napoti na prvo prazno blagajno.
5. Katera strategija (2.,3.ali 4.) daje najboljše, katera pa najslabše rezultate? Za testno besedilo vzemite kakšen niz z interneta.
6. Ponovite naloge od 2 do 5 za tri blagajne. Ko bomo spoznali sezname, bomo nalogo lahko ponovili za k blagajn, kjer je k drug parameter funkcije blagajne(niz,k). Ko bomo spoznali modul random, bomo čase prihodov in čase strežbe lahko določali naključno.