
NAPOJ

# UČNA PRIPRAVA

# Zaporedje

## OSNOVNI PODATKI

| |
|---|
| Šola: |
| Letnik: 1. Letnik |
| Datum: |
| Predmet: Informatika |
| Učna tema: zaporedje |
| Učna enota: zaporedje, tabela, povezan seznam, vrsta, sklad |
| Učne oblike:<br><br>- frontalno, individualno |
| Učne metode:<br><br>- razlaga, pogovor, demonstracija, reševanje problemov |
| Predznanje:<br><br>- poznajo pojme Python funkcija |
| Operativni učni cilji<br><br>Ob koncu učne ure učenec:<br><br>- Razume pojem zaporedja in zakaj je pomembno<br>- Razume podatkovno strukturo tabela<br>- Razume podatkovno strukturo povezan seznam<br>- Razume podatkovno strukturo vrsta<br>- Razume podatkovno strukturo sklad |
| Učna sredstva:<br><br>- Učila: delovni list, računalnik<br>- Učni pripomočki: računalnik, Python, zvezek, tabla |
| Didaktične etape učnega procesa:<br><br>1. Pripravljanje ali uvajanje<br>2. Obravnava nove učne snovi<br>3. Urjenje<br>4. Ponavljanje<br>5. Preverjanje |
| Medpredmetne povezave: slovenščina, angleščina |
| Literatura:<br>- J. Gal-Ezer, T. Vilner, E. Zur, Teaching algorithm efficiency at CS1 level: A different approach, Computer Science Education 14 (3) (2004) 235–248.<br>- M. A. Weiss, Data Structures and Algorithm Analysis (2nd Edition), Addison Wesley, 1994.<br>- M. T. Goodrich, R. Tamassia, Data structures and algorithms in Java, 2001.<br>- G. Anželj, J. Brank, A. Brodnik, P. Bulić, M. Ciglarič, M. Đukić, L. Furst, M. Kikelj, A. Krapež, H. Medvešek, N. Mori, M. Pančur, P. Sterle, Računalništvo in informatika 1, Založba Univerze na Primorskem and Založba Fakultete za računalništvo in informatiko and Založba Fakultete za elektrotehniko and računalništvo in informatiko, 2015. URL https://lusy.fri.uni-lj.si/ucbenik/book/index.html.<br>- T. J. Rolfe, Classroom exercise demonstrating linked list operations, ACM SIGCSE Bulletin 38 (4) |

| |
|---|
| (2006) 83–84.<br>- Chapter 4: Computer organization, Dostopno na: https://www.cs.hmc.edu/csforall/ComputerOrganization/ComputerOrganization.html, (pridobljeno 25. 6. 2018). |
| Novi pojmi:<br>- Zaporedje, povezan seznam, vrsta, sklad. |
| Priloga:<br>- tabela časovne zahtevnosti osnovnih operacij<br>- datoteka povezanSeznam.py<br>- listki z elementi |

# POTEK UČNE URE

## UVODNI DEL: UVAJANJE

| ČAS | UČITELJ | UČENEC | UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI |
|---|---|---|---|
| **Uvod**<br><br>5 min | *Pozdravi učence, zapiše manjkajoče učence in po potrebi prosi dežurnega učenca, da pobriše tablo.*<br><br>Danes se bomo pobližje spoznali z abstraktno podatkovno strukturo zaporedje ter podatkovnimi strukturami tabela, povezan seznam, vrsto in skladom.<br><br>O zaporedju oziroma zaporedju elementov govorimo, ko je vrstni red le-teh pomemben. Eno vrsto zaporedje že poznamo – to je tabela. | Odzdravijo, naštejejo manjkajoče učence, dežurni učenec pobriše tablo.<br><br>Poslušajo. | Frontalno, razlaga. |

## GLAVNI DEL: OBRAVNAVANJE UČNE SNOVI

| VSEBINSKI POUDARKI | UČITELJ | UČENEC | UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI |
|---|---|---|---|
| **Tabela**<br><br>10 min | Tabela je v računalniškem pomnilniku zapisana zelo enostavno – elementi so razporejeni po vrsti, eden za drugim.<br><br>Razmislimo o časovnih zahtevnostih operacij: | Spremljajo razlago in si zapišejo v zvezek. | Frontalno, razlaga, skupinsko delo. |

| | | | |
|---|---|---|---|
| | *Učitelj na tablo nariše tabelo iz priloge 1 in jo ustrezno dopolni.*<br><br>• Dostop to poljubnega elementa: časovna zahtevnost je O(1), saj točno vemo kje v pomnilniku je poljuben indeks.<br>• Dodajanje elementa na koncu: je spet O(1), saj vemo kateri je naslednji neuporabljen indeks.<br>• Vstavljanje elementa: vstavljanje pomeni dodajanje na nekončno pozicijo (tj. vse pozicije v zaporedju razen končnega). Tu pride do težav. Če želimo vstaviti element »30« na 1. mesto, kaj moramo narediti z elementi od prvega do zadnjega mesta? Elemente moramo premakniti za 1 mesto naprej (v desno). Pri časovni zahtevnosti nas, kot vemo, zanima koliko časa potrebuje v najslabšem primeru. Kateri primer je to? Najslabši primer je, ko vstavimo element na prvo mesto (indeks 0), pri čemer premaknemo vseh *n* elementov. Zaradi tega imamo časovno zahtevnost O(n)<br>• Brisanje elementa: tako kot pri dodajanju imamo O(1), le da tu element izbrišemo in ne dodamo.<br>• Odvzemanje elementa: Enako kot dodajanje oziroma vstavljanje. O(1) za brisanje zadnjega elementa, O(n) za brisanje nekončnega elementa, saj moramo v tem primeru premakniti ostale elemente v levo.<br>• Število elementov: koliko elementov ima tabela ima Python že zapisano in ažurirano, zato je časovna zahtevnost O(1)<br><br>Tabela pa ima eno težavo. Kaj se zgodi, ko želimo dodajati elemente v polno tabelo? V pomnilniku moramo ustvariti novo, večjo tabelo in v njo kopirati vse elemente. Kar pa traja veliko časa.<br><br>Tu vstopi podatkovna struktura povezan seznam. | | |
| **Povezan seznam**<br><br>30 minut | Povezan seznam ni nič drugega kot po vrsti povezani podatki oziroma vozlišča. Vsako vozlišče vsebuje podatek ter kazalec na naslednje vozlišče v zaporedju. Tu ni nujno, da so vozlišča v računalniškem pomnilniku shranjena ena za drugo. Za lažje upravljanje imamo še posebno vozlišče na | Spremljajo razlago in si zapišejo v zvezek. | Frontalno, individualno, razlaga, pogovor. |

| | | | |
|---|---|---|---|
| | začetku imenovano glava, ki vsebuje le število elementov v seznamu, kazalec na prvo ter kazalec na zadnje vozlišče.<br><br>Da bo vse skupaj bolj zanimivo, sem pripravil igrico, kjer bomo z vašo pomočjo dodajali hrano na nakupovalni listek.<br><br>V roki imam listke (*v prilogi 3*) ki predstavljajo vozlišča, na katerih sta podatka »hrana« ter »naslednje vozlišče«, ki je na začetku prazno. Na tablo bom sedaj narisal glavo seznama.<br><br>*Učitelj na tablo nariše vozlišče.*<br><br>Recimo, da želim v nakupovalni seznam dodati zelje. Vse kar sedaj naredim je, da podam ta listek enemu od vas *(učitelj poda listek učencu)* in popravim podatke v glavi seznama na tabli.<br><br>*Učitelj na tabli spremeni število elementov v 1, oba kazalca pa na novo narisano vozlišče »zelje«.*<br><br>Sedaj imamo povezan seznam z enim elementom, kazalca v glavi pa kažeta na isto vozlišče, saj je to vozlišče prvo in zadnje hkrati. Dodajmo še eno hrano, nato pa mi boste nadaljevali vi.<br><br>*Učitelj poda listek poljubnemu učencu.*<br><br>Katere podatke moramo sedaj posodobiti?<br><br>*Učitelj ustrezno popravi povezan seznam na tabli. Po želji doda še 1 ali 2 elementa.* | Odgovorijo: v glavi seznama število elementov na 2, kazalec na zadnjo vozlišče naj kaže na novo vozlišče z zadnje dodano hrano, ter pod naslednji element na listku prvega učenca napišemo hrano naslednjega učenca. | |
| | Napolnili smo naš povezan seznam s hrano. Kaj se zgodi s seznamom, če želimo odstraniti zadnji element in kaj, če želimo odstraniti nekončni element? | Odgovorijo: V glavi seznama popravimo število elementov, kazalec na zadnji element in kazalec v prejšnjem elementu. Če odstranimo prvi element, popravimo število | |

<table>
<tr>
<td></td>
<td>Kakšne mislite da so časovne zahtevnosti naslednjih operacij?<br><br>1. Število elementov.<br>2. Dostop do elementa.<br>3. Dodajanje.<br>4. Vstavljanje.<br>5. Brisanje.<br>6. Odvzemanje.<br><br>Učitelj dopolni tabelo iz priloge 1.</td>
<td>elementov in kazalec na prvi element.<br><br>Odgovorijo:<br><br>1. O(1) saj jo imamo zapisano v glavi seznama<br><br>2. O(n), saj moramo iti po vrsti do n-tega vozlišča<br><br>3. O(1), saj imamo kazalec na zadnje vozlišče<br><br>4. O(1), saj samo ažuriramo kazalce (plus čas. zahtevnost iskanja pozicije)<br><br>5. O(1), saj imamo kazalec na zadnje vozlišče<br><br>6. O(n), saj moramo najti prejšnje vozlišče, da lahko prevežemo</td>
<td></td>
</tr>
<tr>
<td>Vrsta<br><br>20 min</td>
<td>Vrsta je tip zaporedja, kjer se elementi dodajajo samo na začetku, odvzemajo pa samo na koncu zaporedja. Tako torej, kot bi si predstavljali iz realnega življenja: recimo vrsta v menzi, kjer učenci vstopajo v vrsto na začetku, izstopajo pa na koncu. Vrsto lahko implementiramo s tabelo ali s povezanim seznamom. Iz strani časovne zahtevnosti si oglejmo, kateri pristop je boljši:<br><br>
<table>
<tr><td>Operacija</td><td>tabela</td><td>Povezan seznam</td></tr>
<tr><td>Dodajanje</td><td>O(1)</td><td>O(1)</td></tr>
<tr><td>Odvzemanje</td><td>O(n)</td><td>O(1)</td></tr>
</table>
</td>
<td>Sledijo razlagi in si zapisujejo.<br><br>Odgovorijo: povezan seznam.</td>
<td>Frontalno, individualno, razlaga, pogovor, demonstracija.</td>
</tr>
</table>

| | | | |
|---|---|---|---|
| | Katero strukturo je bolje uporabiti?<br><br>Povezan seznam sem implementiral že sam, tako da moramo mi le implementirati vrsto samo. Potrebujemo torej napisati funkcije za ustvarjanje nove vrste, dodajanje elementa v vrsto in brisanja elementa. Za pomoč imamo še funkciji jePrazna() ter izpisi(), kjer prva preveri, ali je vrsta prazna, druga pa vrsto izpiše.<br><br>Datoteko povezanSeznam.py si poberite iz spletne učilnice na namizje, tam pa ustvarite novo datoteko imenovano vrsta.py, v kateri bomo sedaj implementirali vrsto.<br><br>`import povezaniSeznam as sez`<br><br>`def ustvari():`<br>`  return sez.Ustvari()`<br><br>`def dodaj(vrsta, element):`<br>`  sez.DodajNaKonec(vrsta, element)`<br><br>`def odvzemi(vrsta):`<br>`  indeks = sez.IndeksZacetnega(vrsta)`<br>`  element = sez.Element(vrsta, indeks)`<br>`  sez.IzlociZacetnega(vrsta)`<br>`  return element`<br><br>`def jePrazna(vrsta):`<br>`  return (sez.SteviloElementov(vrsta) == 0)`<br><br>`def izpisi(vrsta):`<br>`  print(sez.Zaporedje(vrsta))`<br><br>To je vse. Dajmo še preveriti, ali deluje vse v redu.<br><br>`vrsta = ustvari()`<br>`print('Dodamo A v vrsto.')`<br>`dodaj(vrsta, 'A')`<br>`izpisi(vrsta)`<br>`print('Dodamo B v vrsto.')`<br>`dodaj(vrsta, 'B')`<br>`izpisi(vrsta)`<br>`if jePrazna(vrsta):`<br>`  print('Vrsta je prazna.')`<br>`else:`<br>`  print('Vrsta ni prazna.')`<br>`print('Odvzamemo element iz vrste.')`<br>`odvzemi(vrsta)`<br>`izpisi(vrsta)`<br>`print('Odvzamemo element iz vrste.')`<br>`odvzemi(vrsta)`<br>`izpisi(vrsta)`<br>`if jePrazna(vrsta):`<br>`  print('Vrsta je prazna.')`<br>`else:`<br>`  print('Vrsta ni prazna.')` | Si poberejo Python datoteko in ustvarijo datoteko vrsta.py | |

<table>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Sklad<br><br>20 minut</td><td>Sklad je, po domače rečeno, kup elementov, pri čemer lahko dodajamo in odvzemamo samo iz vrha tega kupa. Primer je kup krožnikov, kjer dodajamo in odvzemamo zgolj iz vrha.<br><br>Sklad prav tako lahko implementiramo s tabelo ali s povezanim seznamom. Iz strani časovne zahtevnosti mi povejte, kateri pristop je boljši.<br><br>
<table>
<tr><td></td><td>Tabela</td><td>Povezan seznam</td></tr>
<tr><td>Dodajanje</td><td>O(1)</td><td>O(1)</td></tr>
<tr><td>Odvzemanje</td><td>O(1)</td><td>O(1)</td></tr>
</table>
<br>Ker smo pri vrsti uporabili povezan seznam, uporabimo sedaj tabelo. Sprogramirati moramo iste funkcije kot pri vrsti. Ker pa že znate delati s tabelami, vas prosim, da mi pri tem tudi pomagate.<br><br>
<pre>
def ustvari():
    return []

def dodaj(sklad, element):
    sklad.append(element)

def odvzemi(sklad):
    return sklad.pop()

def jePrazen(sklad):
    return (len(sklad)==0)

def izpisi(sklad):
    print(sklad)
</pre>
<br>Dajmo še preveriti pravilnost delovanja:<br><br>
<pre>
sklad = ustvari()
print('Dodamo A v sklad ...')
dodaj(sklad, 'A')
izpisi(sklad)
print('Dodamo B v sklad ...')
dodaj(sklad, 'B')
izpisi(sklad)
if jePrazen(sklad):
  print('sklad je prazen.')
else:
  print('sklad ni prazen.')
print('Odvzamemo element iz sklada ...')
odvzemi(sklad)
izpisi(sklad)
print('Odvzamemo element iz sklada ...')
vzemi(sklad)
izpisi(sklad)
</pre></td><td>Poslušajo, si zapisujejo.<br><br>Odgovorijo: vseeno, saj so časovne zahtevnosti enake.<br><br>Pomagajo učitelju napisati Python funkcije.</td><td></td></tr>
</table>

| | | | |
|---|---|---|---|
| | **if** jePrazen(sklad):<br>  print('**sklad je prazen.**')<br>**else**:<br>  print('**sklad ni prazen.**') | | |

## ZAKLJUČNI DEL: ZAKLJUČNO PONAVLJANJE / PREVERJANJE

| ČAS | UČITELJ | UČENEC | UČNE OBLIKE, METODE, TEHNIKE, UČNI PRIPOMOČKI |
|---|---|---|---|
| 5 min | Tako, spoznali smo abstraktno podatkovno strukturo zaporedje, katero lahko implementiramo ali s tabelo ali z povezanim seznamom. Sklad in vrsta sta primera zaporedja, kjer se elementi hranijo najnovejšega do najstarejšega za sklad, ter najstarejšega do najnovejšega za vrsto.<br><br>*Se zahvali učencem za sodelovanje, jim da napotke za domačo nalogo. Dežurni učenec naj po potrebi pobriše tablo. Učenci naj ugasnejo računalnike.* | Poslušajo, ugasnejo računalnike, dežurni učenec pobriše tablo. | Frontalno. |

# Priloge

## Priloga 1: tabela časovne zahtevnosti osnovnih operacij

| Operacija | Tabela | Povezan seznam |
|---|---|---|
| Število elementov | O(1) | |
| Dostop do elementa | O(1) | |
| Dodajanje | O(1) | |
| Vstavljanje | O(n) | |
| Brisanje | O(1) | |
| Odvzemanje | O(n) | |

## Priloga 2: datoteka povezanSeznam.py

```
def Ustvari():
  return [[0, 0, 0]]

def SteviloElementov(povSeznam):
  return povSeznam[0][0]

def IndeksZacetnega(povSeznam):
  return povSeznam[0][1]

def IndeksNaslednika(povSeznam, indeks):
  return povSeznam[indeks][1]

def IndeksKoncnega(povSeznam):
  return povSeznam[0][2]

def IndeksPredhodnika(povSeznam, indeks):
  return povSeznam[indeks][2]

def Element(povSeznam, indeks):
  return povSeznam[indeks][0]

def DodajNaKonec(povSeznam, element):
  iPrejKoncnega = povSeznam[0][2]
  iDodanega = len(povSeznam)
  povSeznam.append([element, 0, iPrejKoncnega])
  povSeznam[iPrejKoncnega][1] = iDodanega
  povSeznam[0][2] = iDodanega
  povSeznam[0][0] += 1

def Vstavi(povSeznam, indeks, element):
  iNasled = indeks
  iPredhod = povSeznam[indeks][2]

  iDodanega = len(povSeznam)
  povSeznam.append([element, iNasled, iPredhod])

  povSeznam[iPredhod][1] = iDodanega
  povSeznam[iNasled][2] = iDodanega

  povSeznam[0][0] += 1
```

```
def DodajNaZacetek(povSeznam, element):
  iZacetnega = IndeksZacetnega(povSeznam)
  Vstavi(povSeznam, iZacetnega, element)

def Izloci(povSeznam, indeks):
  iNasled = povSeznam[indeks][1]
  iPredhod = povSeznam[indeks][2]

  povSeznam[iPredhod][1] = iNasled
  povSeznam[iNasled][2] = iPredhod

  povSeznam[0][0] -= 1

def IzlociZacetnega(povSeznam):
  Izloci(povSeznam, IndeksZacetnega(povSeznam))

def IzlociKoncnega(povSeznam):
  Izloci(povSeznam, IndeksKoncnega(povSeznam))

def Zaporedje(povSeznam):
  indeks = IndeksZacetnega(povSeznam)
  elementi = []
  while indeks != 0:
    elementi.append(Element(povSeznam, indeks))
    indeks = IndeksNaslednika(povSeznam, indeks)
  return elementi

def ZaporedjeNazaj(povSeznam):
  indeks = IndeksKoncnega(povSeznam)
  elementi = []
  while indeks != 0:
    elementi.append(Element(povSeznam, indeks))
    indeks = IndeksPredhodnika(povSeznam, indeks)
  return elementi

def IzpisiVse(povSeznam):
  print(povSeznam)
  opis = 'Zaporedje od začetka do konca: '
  print(opis + str(Zaporedje(povSeznam)))
  opis = 'Zaporedje od konca do začetka: '
  print(opis + str(ZaporedjeNazaj(povSeznam)))
```

## Priloga 3: listki z elementi

Element: Jabolko

Naslednji element: _____

Element: Banana

Naslednji element: _____

Element: Korenje

Naslednji element: _____

Element: Napolitanke

Naslednji element: _____

Element: Prepečenec

Naslednji element: _____

Element: Čokolada

Naslednji element: _____

Element: Grozdje

Naslednji element: _____

Element: Pašteta

Naslednji element: _____

Element: Sladoled

Naslednji element: _____

Element: Riba

Naslednji element: _____

Element: Kivi

Naslednji element: _____

Element: Solata

Naslednji element: _____

Element: Puding

Naslednji element: _____

Element: Krompir

Naslednji element: _____

Element: Olive

Naslednji element: _____

Element: Paprika

Naslednji element: _____