Klasični algoritmi za urejanje v bločnem programiranju

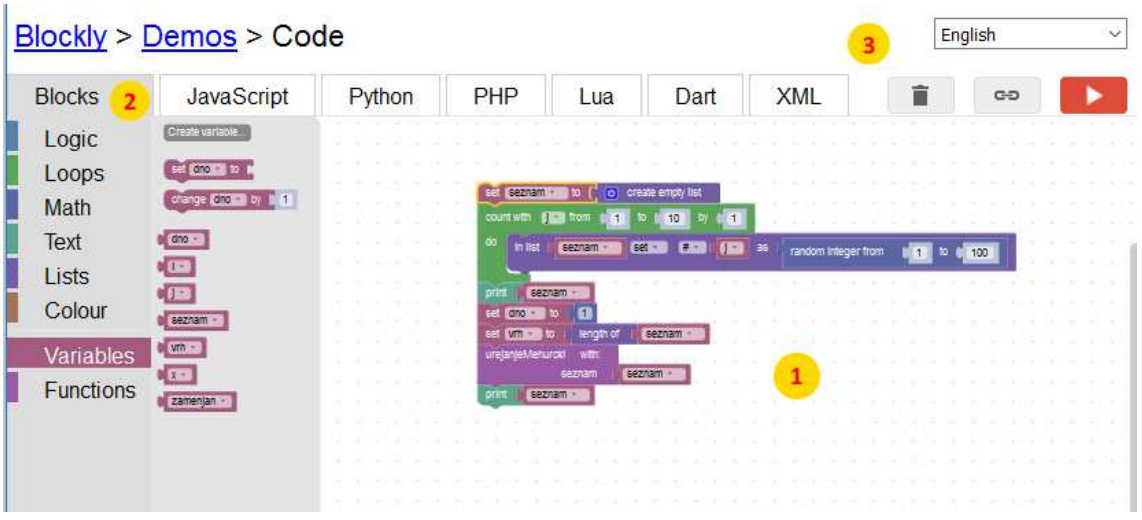
↓↓↓
Igor Pesek

→ V prvi številki letošnjega Preseka smo predstavili program Blockly Games [?], s katerim lahko s pomočjo iger napravite prve korake v programiranje. V današnjem prispevku bomo predstavili spletni program Blockly [?], ki omogoča programiranje v vizualnem okolju s pomočjo blokov. Spletni naslov programa najdete na koncu prispevka med viri. Blockly nima v naprej pripravljenih nalog, kot je to bilo narejeno v Blockly Games, zato bomo delo v okolju Blockly predstavili s pomočjo algoritmov za urejanje.

Priprava seznama za urejanje

Preden se lotimo algoritmov urejanja, potrebujemo seznam naključno izbranih števil, ki jih bomo kasneje uredili. Postopek priprave seznama bomo uporabili tudi za razlago okolja Blockly. Delovno okolje Blockly je prikazano na sliki 1 in je razdeljeno na tri področja. Prvo, označeno s številko 1, je območje, kjer programiramo oz. sestavljamo programe. Območje 2 vsebuje vse bloke, ki jih lahko pri programiranju uporabimo. Bloki so razvrščeni v kategorije po njihovi namembnosti. Ko kakšen blok potrebujemo, ga povlečemo na območje 1. Na območju 3 pa so zavihki, ki prikažejo kodo izbranega programskega jezika; ta je ekvivalentna kodi našega bločnega programa. Skrajno desno je tudi rdeč gumb, ki začne izvajanje našega programa. Če niste reševali Blockly games, si lahko za delo s programom Blockly pomagate s priročnikom, ki ga najdete na `https://github.com/google/blockly/wiki` [?].

Da ustvarimo seznam desetih naključnih števil, moramo najprej ustvariti spremenljivko, ki na začetku predstavlja prazen seznam. Sledi zanka, ki se izvede desetkrat in v vsaki ponovitvi naključno izbere število iz določenega obsega (v našem primeru med 1 in 100). Seznam smo s tem ustvarili, preostane nam še samo klic funkcije za urejanje. Celoten program je prikazan na sliki 2.



SLIKA 1. Delovno okolje programa Blockly

Preden nadaljujemo, bomo predstavili še funkcijo za zamenjavo dveh elementov v seznamu, saj jo pri urejanju večkrat potrebujemo in je zato smiselno, da jo sami sprogramiramo. Na sliki 3 je prikazan postopek za zamenjavo, kjer smo uporabili dodatno spremenljivko. Samostojno razmislite, kako zamenjati dva elementa v seznamu brez uporabe dodatne spremenljivke ali razširitve polja.

Urejanje z mehurčki

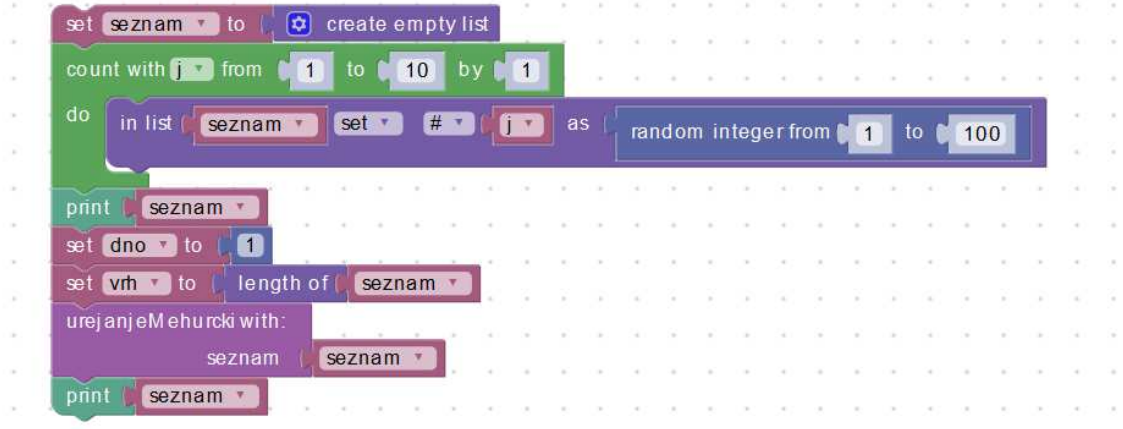
Algoritem urejanja z mehurčki (ang. Bubble sort) je eden najbolj znanih algoritmov za urejanje. Osnovna ideja algoritma je, da pregledujemo seznam od prvega elementa in paroma primerjamo sosedne elemente. Če je levi element para večji od desnega elementa, potem ju zamenjamo. Primerjanje nadaljujemo do konca seznama in postopek ponavljamo, dokler v celotnem seznamu nismo zamenjali nobenega elementa več. Zakaj pa ime urejanje z mehurčki? Ime izhaja iz dejstva, da izgleda tako, kot da levi element para damo v mehurček in ga premikamo desno, dokler je trenutni desni element manjši od elementa v mehurčku. Pomembno je omeniti, da element, ki ga premikamo v mehurčku, ne pride nujno na svoje kočno mesto v urejenem seznamu, torej je lahko večkrat v mehurčku. Časovna zahtevnost algoritma je $\mathcal{O}(n^2)$, saj se v najslabšem primeru zunanja zanka izvede n-krat, notranja pa se vedno izvede $n-1$ krat.

```
UrejanjeZMehurcki(A, dno, vrh):
  zamenjan = true
  ponavljaj dokler zamenjan:
    zamenjan = false
    za vsak i = dno+1 do vrh ponavljaj:
      če (A(i-1) > A(i)) potem
        zamenjaj(A, i-1, i)
        zamenjan = true
```

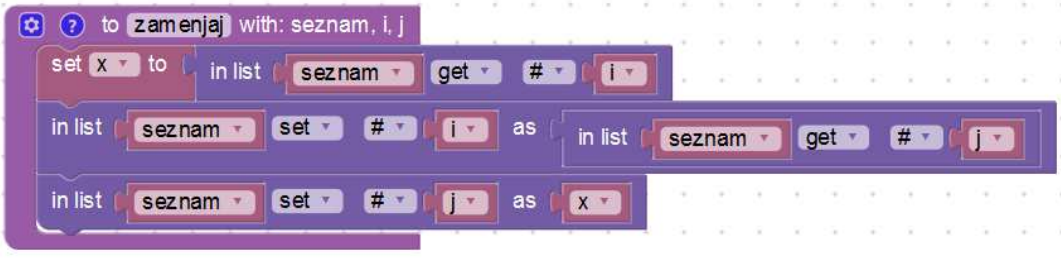
Na sliki 4 je prikazan algoritem *Urejanja z mehurčki* še v Blockly-ju.

Hitro urejanje

Hitro urejanje (ang. QuickSort) [?] je predstavnik algoritmov strategije *Deli in vladaj*, o kateri smo v Pre-



SLIKA 2. Priprava seznama



SLIKA 3. Zamenjava dveh elementov seznama

seku že pisali [?]. Algoritem deluje rekurzivno na seznamu A v dveh korakih.

Deli. Preuredimo seznam $A[\text{dno}, \text{vrh}]$ v dva seznama $A[\text{dno}, q-1]$ in $A[q+1, \text{vrh}]$ na takšen način, da so vsi elementi v seznamu $A[\text{dno}, q-1]$ manjši ali enaki od $A[q]$ ter da so vsi elementi $A[q+1, \text{vrh}]$ večji od $A[q]$. Index q določimo med postopkom preurejanja.

Vladaj. Rekurzivno uredimo podseznama $A[\text{dno}, q-1]$ in $A[q+1, \text{vrh}]$.

Pri strategiji deli in vladaj je običajno še tretji korak, kjer delne rešitve, ki smo jih v koraku deli razdelili, združimo. Pri algoritmu hitro urejanje ta korak ni potreben, saj je delna rešitev že urejena in na pravem mestu.

Najprej bomo zapisali algoritem za korak vladaj.

```
HitroUredi(A, dno, vrh):
  če dno < vrh potem
    q = deli(A, dno, vrh)
    HitroUredi(A, dno, q-1)
    HitroUredi(A, q+1, vrh)
```

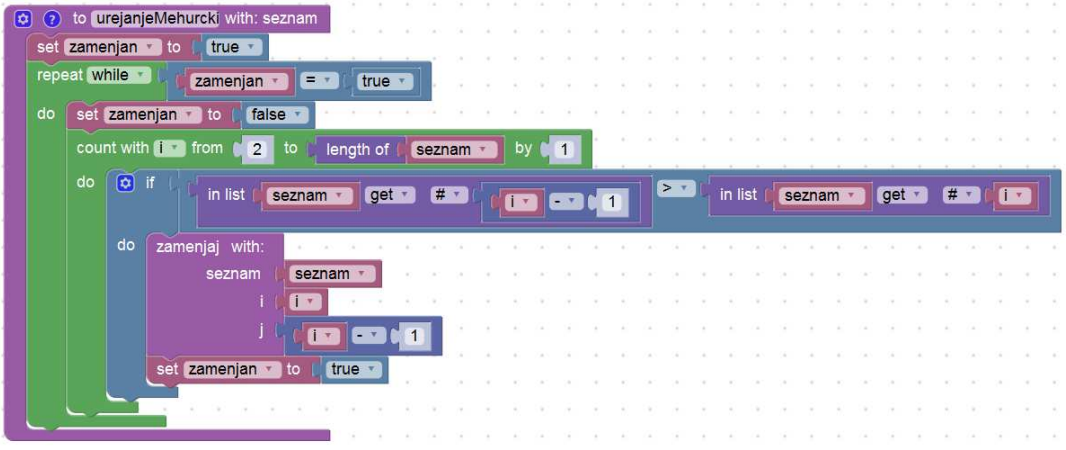
Opazimo, da v rekurzivnih klicih HitroUredi ne urejamo elementa, ki je na q mestu v seznamu A. Zakaj? To je posledica metode deli, ki jo bomo zapisali v nadaljevanju.

```
deli(A, dno, vrh):
  pivot = A[dno]
  i = dno
  j = vrh
  ponavljaj dokler (i < j):
    ponavljaj dokler (A[i] <= pivot):
      i++
    ponavljaj dokler (A[j] > pivot):
      j++
    če (i < j) potem:
      zamenjaj(A,i,j)


  če (i > j) potem:
    zamenjaj(A,dno,j)

  rezultat = j
```

Na začetku metode določimo pivotni element, običajno je to prvi ali zadnji element seznama; v našem primeru je to prvi. S pomočjo pivota bomo seznam A razdelili na dva podseznama. V prvem seznamu bodo elementi, ki so manjši od pivota, v drugem pa elementi seznama A, ki so večji od pivota. Prvi seznam gradimo od indeksa dno proti vrhu (od leve proti desni glede na seznam), drugi seznam pa gradimo od vrha proti dnu (od desne proti levi). Sam algoritem za delovanje ne potrebuje dodatnega pomnilniškega prostora, saj neposredno preureja originalni seznam. Z zankami *ponavljaj* sedaj pregledujemo polje z leve in desne ter iščemo elemente,



SLIKA 4. Urejanje z mehurčki

- ki so večji od pivotnega elementa (prva notranja zanka *ponavljaj*) in
- ki so manjši od pivotnega elementa (druga notranja zanka *ponavljaj*).

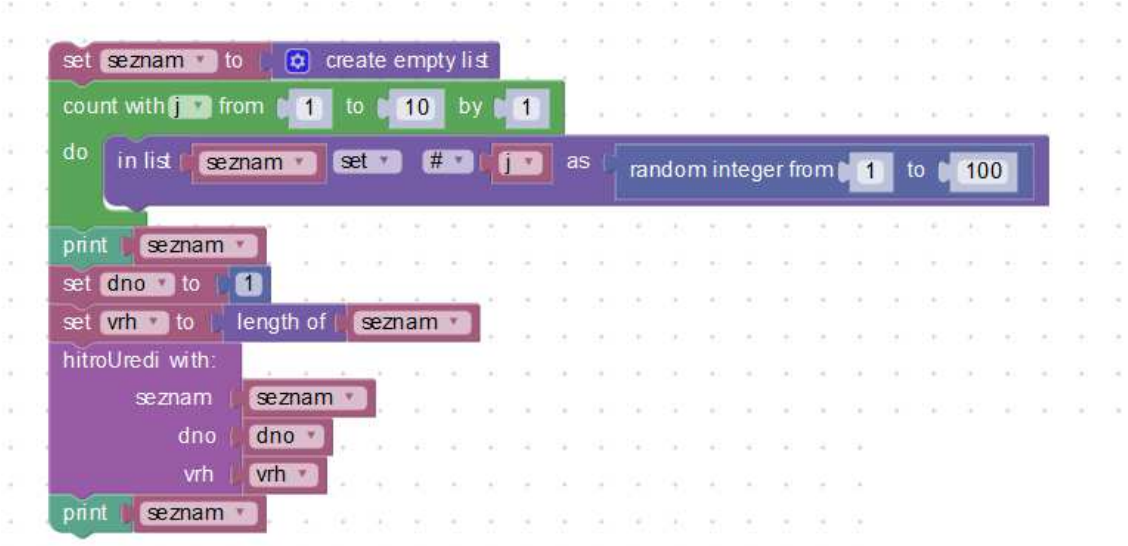
Ko se obe zanki zaključita, sta možni dve situaciji in sicer,

- da je i manjši od j ali
- da je i večji od j.

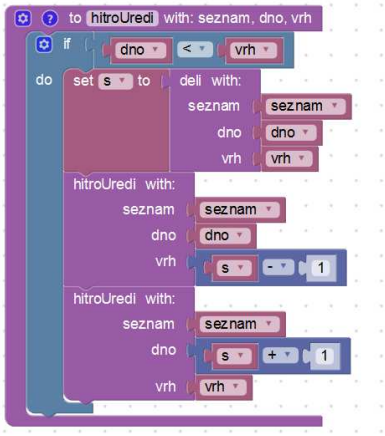
V prvem primeru imamo še vedno dva elementa, kjer je element na i-tem mestu večji od pivota in element na j-tem mestu manjši od pivota. Zato ju lahko takoj zamenjamo, saj na ta način na začetek seznama prestavljamo manjše elemente in na konec večje elemente. V drugem primeru pa se indeksa i in j prekrižata, kar pomeni, da nismo našli več takšnega para elementov, ki bi na obeh indeksih stali na napačnih mestih. Zato menjave ne izvedemo, zaključimo pa zunanjo zanko ponavljaj, saj s trenutnim pivotom ne moremo več deliti seznama. V zadnjem koraku moramo umestiti pivotni element v seznam na tako mesto, da bodo levo od njega manjši elementi in desno od njega večji. Katero mesto v seznamu je to? Razmislek pokaže, da je to lahko ali indeks i ali indeks j. Če bi menjali element na indeksu i, potem bi na indeks dno postavili element, ki je večji od pivota (element na i-tem indeksu namreč to je). Zaradi tega bi prišli v navzkrižje s pravilom, da so v levem podseznama vsi elementi manjši od pivota. Če menjamo pivot z j-tim elementom (ta kaže na element, ki je manjši od pivota), pravila ne kršimo, zato menjamo pivot in j-ti element. Posledično pivotni element sedaj stoji na mestu, ki se do konca urejanja ne bo več spremenilo, saj so v seznamu levo od njega manjši in desno večji elementi in ga zato ni potrebno več urejati niti upoštevati pri rekurzivnih klicih. To je tudi odgovor na prej zastavljeno vprašanje, zakaj elementa na mestu q ne urejamo več.

Časovna zahtevnost algoritma je v najslabšem primeru $\mathcal{O}(n^2)$, vendar se je v testih pokazalo, da je praktična časovna zahtevnost $\mathcal{O}(n \log n)$, kar ga uvršča med najhitrejše algoritme urejanja. Njegova prednost je tudi, da ne zahteva dodatnega prostora, saj celotno urejanje poteka samo znotraj danega seznama. Algoritem Hitro urejanje bomo implementirali tudi v vizualnem okolju Blockly. Kot smo to storili že pri urejanju z mehurčki, najprej z naključnimi števili napolnimo seznam (slika 5).

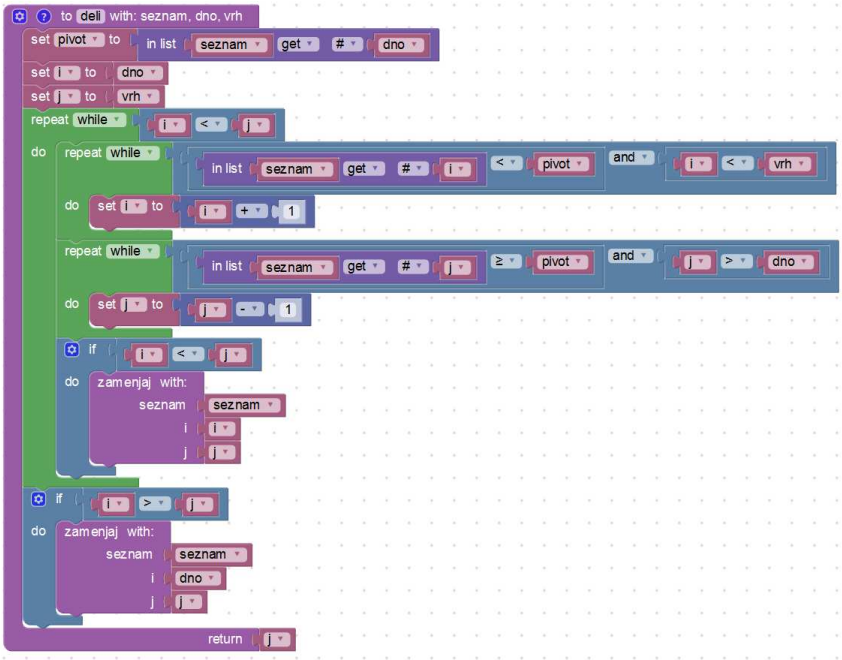
Nato sledi klic funkcije HitroUredi, ki je prikazana na sliki 6. Ker funkcijo uporabljamo rekurzivno, je prisoten zaustavitveni pogoj, kjer preverimo, ali je potrebno urediti še kakšen podseznam. V primeru, da je dno = vrh, bi urejali samo en element, kar pa ni smiselno.



SLIKA 5.
Pripravimo seznam



SLIKA 6.
Hitro uredi



SLIKA 7.
Postopek deli

V pogojnem stavku najprej pokličemo funkcijo deli, ki seznam na prej opisan način preuredi in kot rezultat vrne na mesto, kjer je pivotni element. Funkcija deli je prikazana na sliki 7.

Kot smo že uvodoma zapisali, nam okolje Blockly omogoča, da algoritem, ki smo ga bločno sprogramirali, izpišemo tudi v različnih programskih jezikih. V nadaljevanju je na sliki 8 prikaz algoritma Hitro urejanje v programskem jeziku Python. Opazimo, da je sedaj razumevanje kode zapisane v Pythonu lažje, če ga lahko primerjamo z Blockly različico.

Zaključek

V prispevku smo predstavili program Blockly, ki omogoča bločno programiranje tudi bolj zahtevnih algoritmov. Razložili smo delovanje dveh algoritmov za urejanje, urejanje z mehurčki ter hitro urejanje in jih sprogramirali v Blockly-ju. Velika prednost Blockly-ja pri učenju programiranja je, da uporabniku omogoča program, izdelan v bločnem načinu izpisati v različnih programskih jezikih, kar lahko koristi pri učenje izbranega programskega jezika.

Literatura

[1] I. Pesek, *Naučimo se programiti s pomočjo vizualnega programiranja*, Presek 44 (1), 2016/2017.

[2] `https://blockly-demo.appspot.com/static/demos/code/`, ogled 20. 11. 2016.

[3] T. H. Cormen et al., *Introduction to Algorithms*, MIT Press, 2009.

[4] `https://github.com/google/blockly/wiki`, ogled 20. 11. 2016.

[5] A. Taranenko, *Reševanje problemov s pristopom deli in vladaj*, Presek 37 (4), 2009/2010.

```
import random

x = None
j = None
seznam = None
i = None
zamenjan = None
vrh = None
dno = None
pivot = None
s = None
list2 = None

"""Funkcija v seznamu zamenja i-ti in j-ti element
"""
def zamenjaj(seznam, i, j):
  x = seznam[int(i - 1)]
  seznam[int(i - 1)] = seznam[int(j - 1)]
  seznam[int(j - 1)] = x

"""Funkcija z metodo Hitro Uredi uredi dano
  zaporedje"""
def hitroUredi(seznam, dno, vrh):
  if dno < vrh:
    s = deli(seznam, dno, vrh)
    hitroUredi(seznam, dno, s - 1)
    hitroUredi(seznam, s + 1, vrh)
```

```
"""Funkcija preuredi seznam v dva manjša seznama,
za katera velja, da so elementi levega seznama
manjši od pivota ter elementi desnega seznama
večji od pivota.vrne mesto na katerem je
postavljen pivotni element."""

def deli(seznam, dno, vrh):
  pivot = seznam[int(dno - 1)]
  i = dno
  j = vrh
  while i < j:
    while seznam[int(i - 1)] <= pivot and i < vrh:
      i = i + 1
    while seznam[int(j - 1)] > pivot and j > dno:
      j = j - 1
    if i < j:
      zamenjaj(seznam, i, j)
  if i > j:
    zamenjaj(seznam, dno, j)
  return j

seznam = []
for j in range(1, 11):
  seznam[int(j - 1)] = random.randint(1, 100)
print(seznam)
dno = 1
vrh = len(seznam)
hitroUredi(seznam, dno, vrh)
print(seznam)
```

SLIKA 8.
Hitro urejanje v Pythonu

× × ×

↓ ↓ ↓

REŠITEV NAGRADNE KRIŽANKE

PRESEK 44/2

→ Pravilna rešitev nagradne križanke iz druge številke 44. letnika Preseka je **Holditchev izrek**. Izmed pravilnih rešitev so bili izžrebani URŠKA POJE iz Ljubljane, TADINA BENCE VIRAG iz Lendave in LIDIJA MIKEC iz Novega mesta, ki so razpisane nagrade prejeli po pošti.

× × ×