PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **15** (1987/1988)
Številka 5
Strani 275–279

Matija Lokar:

PODATKOVNE STRUKTURE – VRSTA

Ključne besede: matematika, računalništvo, podatkovne strukture, vrsta.

Elektronska verzija: http://www.presek.si/15/909-Lokar.pdf

PODATKOVNE STRUKTURE – VRSTA

Kaja, Žiga in Nejc so se odpravili v kino.

"Ah, že spet vrsta!" zavzdihne Kaja, ko zagleda kačo ljudi, ki se vije pred blagajno.

"Ostali bomo brez kart," se ustraši Žiga. Zato se Nejc odpravi kar proti blagajni in se meni nič tebi nič postavi pred dekleti, ki ga, zaverovani v klepet, sploh ne opazita.

"Mi je le uspelo," se veseli. A prezgodaj! Nekdo ga potreplja po rami in mu pokaže, kje je konec vrste.

Kot v vsakdanjem življenju moramo pogosto tudi v računalništvu vzdrževati določen red. Tako podatke vodimo v podatkovni strukturi. V zadnji številki lanskoletnega Preseka smo se že spoznali z eno od osnovnih struktur – s skladom. Ta bi našim trem prijateljem verjetno prej kar prav prišel, saj tam velja načelo "zadnji noter, prvi ven". Vendar moramo biti včasih tudi pri programiranju "pošteni" in spoštovati, da kdor prej pride, prej melje. Zato si oglejmo novo podatkovno strukturo, ki to omogoča. Imenovali jo bomo kar **vrsta.** Je zelo podobna skladu, saj sta obe strukturi le posebna primera splošnejše strukture z imenom **urejeni seznam.** Opraviti imamo torej z določenim zaporedjem podatkov. Tudi operacije nad vrsto so podobne operacijam nad skladom. Sprememba je le v vrstnem redu izločanja. Ker smo rekli, da je vrsta "poštena", bo prvi izločen tisti element, ki je v vrsto prišel prvi. Je torej FIFO (**F**irst **I**n **F**irst **O**ut) seznam. Elemente vstavljamo na enem **koncu** vrste in jih jemljemo na drugem (**začetku** vrste). Vrsto elementi zapuščajo v istem vrstnem redu, kot so vanjo prihajali. V nasprotju s skladom ima vrsta torej dva konca, kjer se nekaj dogaja.

Kakor smo formalno predstavili sklad, storimo tako tudi z vrsto.

```
structure vrsta (podatki, Boolean);
(* struktura sklad nad poljubno zalogo vrednosti podatek *)
begin
    declare
        (* pripravi prazno vrsto *)
        pripravi : Ø ⇔ vrsta;
        (* vstavi podatek v vrsto in vrne novo vrsto *)
        vstavi : (podatek, vrsta) ⇔ vrsta;
```

```
        ( * vrne podatek, ki je na začetku vrste * )
        začetek : vrsta ⇔ podatek;
        ( * izbriše prvi element vrste in vrne novo vrsto * )
        odstrani : vrsta ⇔ vrsta;
        ( * ali je vrsta prazna? * )
        prazna : vrsta ⇔ Boolean;
    where
        prazna(pripravi) ::= true;
        prazna(vstavi(p, v)) ::= false;
        odstrani(pripravi) ::= napaka;
        odstrani(vstavi(p, v)) ::=
            if prazna(v) then pripravi else vstavi (p, odstrani( v));
        začetek(pripravi) ::= napaka;
        začetek(vstavi(p, v)) ::=
            if prazna(v) then p else začetek (v);
end.
```

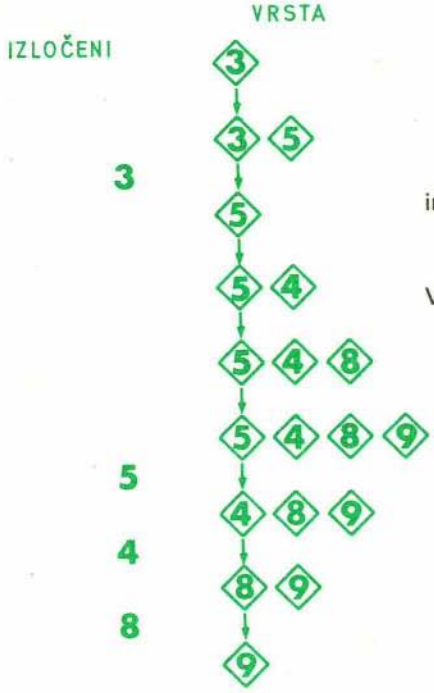
Da nam bodo operacije postale bolj domače, si oglejmo, kako se obnašata sklad in vrsta pri naslednjem zaporedju operacij odstrani in vstavi:

vstavi 3, vstavi 5, odstrani, vstavi 4, vstavi 8, vstavi 9, odstrani, odstrani, odstrani

(glej Sliko 1)

Kako pa je z uporabo vrste? Kot smo že omenili, uporabljamo vrsto tam, kjer želimo podatke obravnavati v istem vrstnem redu, kot jih dobivamo. Tako vrsto uporabljamo pri simulacijah, pri vzdrževanju vrstnega reda izpisovanja datotek na tiskalniku, pri dodeljevanju pomožnega pomnilnika ...

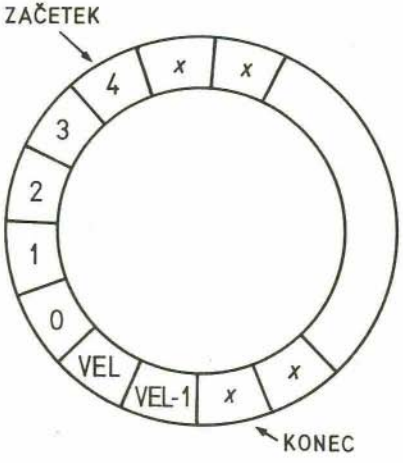
Tudi vrsto bomo predstavili s pomočjo tabele v basicu. Poznati moramo začetek in konec vrste ter vrstni red elementov. Ker se nam na enem koncu vrsta polni, na drugem prazni, mora biti predstavitev taka, da bomo sproščeni prostor z začetka spet uporabili za vstavljanje. Pri tem pa moramo seveda ohraniti pravilni vrstni red. Zato tabelo vodimo krožno, po modulu velikosti. Kaj pa je spet to — po modulu velikosti? Ustrezno mesto za vstavljanje izračunamo tako, da najprej povečamo konec vrste za eno. Nato določimo ostanek pri deljenju z velikostjo vrste. (V pascalu nam to opravi vgrajena funkcija mod, v basicu pa si jo bomo pripravili sami. Seveda mora biti basic, ki ga uporabljamo tak, da šteje polja od 0 dalje. Kaj pa če šteje polja od 1 dalje?) Pri tem naletimo na težavo, kako ločiti med prazno in polno vrsto. Zato se odpovemo enemu mestu v tabeli in z "začetek" označimo raje indeks zadnjega praznega mesta pred začetkom vrste. Tako je vrsta prazna, če je



Slika 1

začetek = konec
in polna, če je
začetek = (konec + 1) mod velikost

V vrsti



Slika 2

si torej slede elementi $v_5, v_6, v_7, v_1, \ldots, v_{velikost-3}, v_{velikost-2}$.

Sestvimo sedaj podprograme:

```
 10 REM
 20 REM pripravi vrsto velikost VELIKOST
 30 REM
 40 VELIKOST = ...
 50 DIM VRSTA (VELIKOST)
 60 ZACETEK = 1 : KONEC = 1
 70 DEF  FN  MOD(A, B) = A – INT(A/B) * B
 80 RETURN

100 REM
110 REM če je vrsta prazna, dobi spremenljivka PRAZNA
120 REM vrednost 1, drugače 0
130 REM
140 PRAZNA = 0
150 IF  ZACETEK = KONEC  THEN  PRAZNA = 1
160 RETURN

200 REM
210 REM vstavi podatek ELEMENT v vrsto
220 REM
230 POM = FN MOD(KONEC + 1, VELIKOST)
235 IF  NOT(ZACETEK = POM)  THEN  GOTO  260
240 REM vrsta je polna – primerno ukrepamo
250 ...
260 REM vrsta ni polna
270 KONEC = POM
280 VRSTA (KONEC) = ELEMENT
290 RETURN

300 REM
310 REM briše podatek iz vrste in ga shrani v ELEMENT
320 REM
330 IF ZACETEK = KONEC  THEN ...: REM napaka, primerno ukrepamo
340 ZACETEK = FN  MOD(ZACETEK + 1, VELIKOST)
350 ELEMENT = VRSTA(ZACETEK)
360 RETURN
```