**Cvetana Krstev**

# Strukture informacija – deo 3

## Nizovi

U većinu programskih jezika su ugrađeni *konstruktori tipova* koji omogućavaju da se, polazeći od elementarnih tipova, konstruišu složeniji tipovi. Osnovni konstruktor tipa je *stepen* koji odgovara operaciji stepenovanja skupa. Operacija stepenovanja se definiše na sledeći način:

$$\mathbf{T}^2 = \{x \mid x = ab, a, b \in \mathbf{T}\}$$

Ovo znači da se kvadrat skupa **T** definiše kao skup koji sadrži elemente nastale dopisivanjem elemenata polaznog skupa. Ovaj konstruktor omogućava da se, polazeći od nekog tipa **T**, konstruiše *niska objekata istog tipa* **T** *konačne dužine* **n**. Ako je **T** neki tip, a ceo broj $n > 0$, tada se $n$-ti stepen tipa **T** novi tip $\mathbf{T}^n$, definiše na sledeći način:

$\mathbf{T}^0 = \varnothing$
$\mathbf{T}^1 = \mathbf{T}$
$\mathbf{T}^{n+1} = \mathbf{T}^n \cdot \mathbf{T}$, za $n > 0$

Elementi tipa $\mathbf{T}^n$, za fiksirani ceo broj $n > 0$, su niske elemenata tipa **T** dužine $n$. U proceduralnim programskim jezicima, kakav je i VBA, pomoću stepena se konstruiše tip koji se naziva *niz* (*array*). Karakteristika nizova u višim programskim jezicima je da je svaki element niza zapravo promenljiva tipa **T** koja se može koristiti u izrazima na isti način kao i svaka druga promenljiva tog tipa. Osim toga, svakom elementu niza može se direktno pristupiti preko *indeksa* niza, pri čemu indeks niza odgovara poziciji elementa u niski iz $\mathbf{T}^n$. Svaki niz tipa **T** se može zamisliti i kao konačna sekvencija, tj. matematička funkcija čiji je domen podskup prirodnih brojeva [0,1,2,..., *n-1*], a opseg skup **T.** U programskom jeziku Python konstrukt „list“ samo približno odgovara konstruktoru tipa niz – svakom elementu liste se može pristupiti preko indeksa, ali svi elementi liste ne moraju biti istog tipa.

**Primer 1.** Sastaviti algoritam koji računa srednju vrednost članova niza i standardnu devijaciju σ.

Neka je $a_1$, $a_2$, $a_3$,..., $a_n$ neki niz od $n$ elemenata (brojeva). Srednja vrednost se računa po formuli $sv = (a_1 + a_2 + a_3 +...+ a_n)/n$. Standardna devijacija je srednje odstupanje vrednosti niza od srednje vrednosti. Računa se kao kvadratni koren srednje vrednosti zbir kvadrata razlika elemenata niza i srednje vrednosti:
$sd = \sqrt{((a_1 - sv)^2 + (a_2 - sv)^2 + (a_3 - sv)^2 +...+ (a_n - sv)^2)/n}$

```
/* računanje srednje vrednosti */
sv ← 0
za i od 1 do n
        sv ← sv + a_i
do ovde i
sv ← sv / n
/* računanje standradne devijacije */
sd ← 0
za i od 1 do n
        sd ← sd + (a_i – sv)^2
do ovde i
sd ← sqrt(sd / n)
```

Ilustracija funkcionisanja algoritma:
Ulaz $n = 8$

| **Element** | 2 | 4 | 4 | 4 | 5 | 5 | 7 | 9 |
|---|---|---|---|---|---|---|---|---|
| **Indeks** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Tok rada algoritma:

| ***i*** | ***sv*** | $a_i$ |
|---|---|---|
| | 0 | |
| 1 | 2 | $a_1 = 2$ |
| 2 | 6 | $a_2 = 4$ |
| 3 | 10 | $a_3 = 4$ |
| 4 | 14 | $a_4 = 4$ |
| 5 | 19 | $a_5 = 5$ |
| 6 | 24 | $a_6 = 5$ |
| 7 | 31 | $a_7 = 7$ |
| 8 | 40 | $a_8 = 9$ |

$sv \leftarrow 40/8 = 5.00$ (srednja vrednost)

| ***i*** | ***sd*** | $a_i$ | $(a_i - sv)^2$ |
|---|---|---|---|
| | 0 | | |
| 1 | 9 | $a_1 = 2$ | $(2-5)^2$ |
| 2 | 10 | $a_2 = 4$ | $(4-5)^2$ |
| 3 | 11 | $a_3 = 4$ | $(4-5)^2$ |

| | | | |
|---|---|---|---|
| 4 | 12 | $a_4 = 4$ | $(4-5)^2$ |
| 5 | 12 | $a_5 = 5$ | $(5-5)^2$ |
| 6 | 12 | $a_6 = 5$ | $(5-5)^2$ |
| 7 | 16 | $a_7 = 7$ | $(7-5)^2$ |
| 8 | 32 | $a_8 = 9$ | $(9-5)^2$ |

$sd \leftarrow \sqrt{32/8} = \sqrt{4} = 2$ (standardna devijacija)

☺

**Primer 2.** Sastaviti algoritam koji izbacuje datu vrednost iz niza – elementi iz repa niza se moraju pomeriti ulevo.
Neka je $a_1, a_2, a_3, ..., a_n$ neki niz od $n$ elemenata, i neka je $x$ vrednost koja se izbacuje. Ovaj algoritam izbacuje iz niza sve elemente čija je vrednost $x$.

$j \leftarrow 0$
**za** $i$ **od** 1 **do** $n$
    **ako** $a_i <> x$ **onda**
        $j \leftarrow j + 1$
        $a_j \leftarrow a_i$
    **kraj ako-onda**
**do ovde** $i$
$n \leftarrow j$

Ilustracija funkcionisanja algoritma:
Ulaz $x = 2$, $n = 8$

| **Element** | 3 | 7 | 2 | 5 | 1 | 2 | 8 | 3 |
|---|---|---|---|---|---|---|---|---|
| **Indeks** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Tok rada algoritma:

| ***i*** | $a_i <> x$ | ***j*** | $a_j \leftarrow a_i$ | $a_1, a_2, a_3, ..., a_n$ |
|---|---|---|---|---|
| | | 0 | | |
| 1 | 3 <> 2 DA | 1 | $a_1 \leftarrow 3$ | **3** 7 2 5 1 2 8 3 |
| 2 | 7 <> 2 DA | 2 | $a_2 \leftarrow 7$ | **3 7** 2 5 1 2 8 3 |
| 3 | 2 <> 2 NE | | | |
| 4 | 5 <> 2 DA | 3 | $a_3 \leftarrow 5$ | **3 7 5** 5 1 2 8 3 |
| 5 | 1 <> 2 DA | 4 | $a_4 \leftarrow 1$ | **3 7 5 1** 1 2 8 3 |
| 6 | 2 <> 2 NE | | | |
| 7 | 8 <> 2 DA | 5 | $a_5 \leftarrow 8$ | **3 7 5 1 8** 2 8 3 |
| 8 | 3 <> 2 DA | 6 | $a_6 \leftarrow 3$ | **3 7 5 1 8 3** 8 3 |

$n \leftarrow 6$ (nova dužina niza)

☺

**Primer 3.** Sastaviti algoritam koji na početak niza ubacuje novi element koji ima zadatu vrednost – elementi počev od repa niza se moraju pomeriti udesno.
Neka je $a_1$, $a_2$, $a_3$,..., $a_n$ neki niz od $n$ elemenata, i neka je $x$ vrednost koja se ubacuje na početak.

**za** $i$ **od** $n$ **do** 1 **korak** -1
    $a_{i+1} \leftarrow a_i$
**do ovde** $i$
$a_1 \leftarrow x$
$n \leftarrow n + 1$
Ilustracija funkcionisanja algoritma:
Ulaz $x = 9$, $n = 5$

| **Element** | 3 | 7 | 2 | 5 | 1 |
|---|---|---|---|---|---|
| **Indeks** | 1 | 2 | 3 | 4 | 5 |

Tok rada algoritma:

| $i$ | $a_{i+1} \leftarrow a_i$ | $a_1, a_2, a_3,..., a_n$ |
|---|---|---|
| 5 | $a_6 \leftarrow a_5$(=1) | 3 7 2 5 1 **1** |
| 4 | $a_5 \leftarrow a_4$(=5) | 3 7 2 5 **5 1** |
| 3 | $a_4 \leftarrow a_3$(=2) | 3 7 2 **2 5 1** |
| 2 | $a_3 \leftarrow a_2$(=7) | 3 7 **7 2 5 1** |
| 1 | $a_2 \leftarrow a_1$(=3) | 3 **3 7 2 5 1** |
| | $a_1 \leftarrow 9$ | **9 3 7 2 5 1** |

$n \leftarrow 6$ (nova dužina niza)
☺

**Primer 4.** Sastaviti algoritam koji od niza čiji su elementi u rastućem redosledu – $a_1$, $a_2$, $a_3$,..., $a_n$ formira niz u kome nema elemenata koji se ponavljaju.
**Rešenje:**
Neka je $a_1 \le a_2 \le a_3 \le ... \le a_n$ neki niz od $n$ elemenata. Treba formirati niz $a_1 < a_2 < a_3 < ... < a_m$ pri čemu je $m \le n$. Koriste se dva brojača; $i$ koji prolazi kroz početne vrednosti niza i $j$ koji pamti pozicije za smeštanje vrednosti novoformiranog niza.

$j \leftarrow 1$
*prethodni* $\leftarrow a_1$
$i \leftarrow 2$
**sve dok** ($i \le n$) **ponavljaj**
**ako** $a_i$ = = *prethodni* **onda**
    $i \leftarrow i + 1$

**inače**

$j \leftarrow j + 1$; $a_j \leftarrow a_i$; $prethodni \leftarrow a_i$; $i \leftarrow i + 1$

**kraj ako-onda**

**do ovde**

$m = j$

Ilustracija funkcionisanja algoritma:

| $a_i$ | 1 | 2 | 2 | 3 | 4 | 4 | 4 | 5 | 5 |
|---|---|---|---|---|---|---|---|---|---|
| **indeks** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

$n = 9$

| $a_i$ | $prethodni = a_i$ ? | $j$ | niz | $prethodni$ | $i$ |
|---|---|---|---|---|---|
| | | 1 | **1** 2 2 3 4 4 4 5 5 | 1 | 2 |
| 2 | F | 2 | **1 2** 2 3 4 4 4 5 5 | 2 | 3 |
| 2 | T | 2 | **1 2** 2 3 4 4 4 5 5 | 2 | 4 |
| 3 | F | 3 | **1 2 3** 3 4 4 4 5 5 | 3 | 5 |
| 4 | F | 4 | **1 2 3 4** 4 4 4 5 5 | 4 | 6 |
| 4 | T | 4 | **1 2 3 4** 4 4 4 5 5 | 4 | 7 |
| 4 | T | 4 | **1 2 3 4** 4 4 4 5 5 | 4 | 8 |
| 5 | F | 5 | **1 2 3 4 5** 4 4 5 5 | 5 | 9 |
| 5 | T | 5 | **1 2 3 4 5** 4 4 5 5 | 5 | 10 |

$m \leftarrow 5$

**Primer 5.** Sastaviti algoritam koji od dva niza čiji su elementi u rastućem redosledu – $a_1$, $a_2$, $a_3$,..., $a_n$ i $b_1$, $b_2$, $b_3$,..., $b_m$ formira treći niz $c$ čiji će elementi biti takođe u rastućem redosledu.

**Rešenje:**

Neka je $a_1 \le a_2 \le a_3 \le ... \le a_n$ neki niz od $n$ elemenata i $b_1 \le b_2 \le b_3 \le ... \le b_m$ neki niz od $m$ elemenata. Polazi se od prvih elemenata oba niza – $a_1$ i $b_1$ čije se vrednosti porede ($i = 1$ i $j = 1$):

- ako je $a_1 < b_1$ onda $a_1$ postaje prvi element niza $c$ ($c_1$), a element $a_2$ postaje sledeći element niza $a$ koji se poredi ($i = 2$);
- ako je $a_1 > b_1$ onda $b_1$ postaje prvi element niza $c$ ($c_1$), a element $b_2$ postaje sledeći element niza $b$ koji se poredi ($j = 2$);
- ako je $a_1 = b_1$ onda $a_1$ i $b_1$ postaju prvi i drugi element niza $c$ ($c_1$ i $c_2$), a elementi $a_2$ i $b_2$ postaju sledeći elementi nizova $a$ i $b$ koji se porede ($i = 2$, $j = 2$);

Ceo postupak se ponavlja dok se ne dođe do kraja prvog, drugog ili oba niza – u tom slučaju se elementi niza koji nije iscrljen prepisuju na kraj niza $c$.

$i \leftarrow 1$
$j \leftarrow 1$
$k \leftarrow 1$
**sve dok** $(i \leq n)$ **I** $(j \leq m)$ **ponavljaj**
**ako** $a_i < b_j$ **onda**
    $c_k \leftarrow a_i$; $i \leftarrow i + 1$; $k \leftarrow k + 1$
**inače ako** $a_i > b_j$ **onda**
    $c_k \leftarrow b_j$; $j \leftarrow j + 1$; $k \leftarrow k + 1$
**inače**
    $c_k \leftarrow a_i$; $i \leftarrow i + 1$; $k \leftarrow k + 1$; $c_k \leftarrow b_j$; $j \leftarrow j + 1$; $k \leftarrow k + 1$
**kraj ako-onda**
**do ovde**
**sve dok** $i < n$ **ponavljaj**
    $c_k \leftarrow a_i$; $i \leftarrow i + 1$; $k \leftarrow k + 1$
**do ovde**
**sve dok** $j < m$ **ponavljaj**
    $c_k \leftarrow b_j$; $j \leftarrow j + 1$; $k \leftarrow k + 1$
**do ovde**

Ilustracija funkcionisanja algoritma:

| $a_i$ | 1 | 2 | 3 | 5 | 7 | 11 | 13 |
|---|---|---|---|---|---|---|---|
| **indeks** | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

| $b_j$ | 4 | 5 | 8 |
|---|---|---|---|
| **indeks** | 1 | 2 | 3 |

| $i$ | $j$ | $k$ | $(i \leq n)$ **I** $(j \leq m)$ | $a_i$ | $b_j$ | $a_i < b_j$ | $a_i > b_j$ | $c_k$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | T ∧ T | 1 | 4 | T | | 1 |
| 2 | 1 | 2 | T ∧ T | 2 | 4 | T | | 2 |
| 3 | 1 | 3 | T ∧ T | 3 | 4 | T | | 3 |
| 4 | 1 | 4 | T ∧ T | 5 | 4 | F | T | 4 |
| 4 | 2 | 5 | T ∧ T | 5 | 5 | F | F | 5 |
| 5 | 2 | 6 | | | | | | 5 |
| 5 | 3 | 7 | T ∧ T | 7 | 8 | T | | 7 |
| 6 | 3 | 8 | T ∧ T | 11 | 8 | F | T | 8 |
| 6 | 4 | 9 | T ∧ F | | | | | |
| | | | | | | | | |

| $i \le n$ | $i$ | $k$ | $a_i$ | $c_k$ |
|---|---|---|---|---|
| T | 6 | 9 | 11 | 11 |
| | 7 | 10 | 13 | 13 |
| $j \le m$ | | | | |
| F | | | | |

| $c_k$ | 1 | 2 | 3 | 4 | 5 | 5 | 7 | 8 | 11 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|
| **indeks** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

☺

**Zadatak.** Dati su sledeći podaci:[1]

Lična primanja i troškovi javnog obrazovanja u zemlji *X* u dolarima

| godina | prihod po stanovniku | troškovi po studentu |
|---|---|---|
| 1930 | 6.610 | 710 |
| 1940 | 6.960 | 950 |
| 1950 | 9.540 | 1.330 |
| 1960 | 12.780 | 2.020 |
| 1970 | 17.340 | 3.440 |
| 1980 | 20.150 | 4.400 |
| 1990 | 24.230 | 5.890 |

Izračunati:

(a) Koliki je približno prosečan (srednji) rast primanja po glavi stanovnika po godini u zemlji *X* od 1930 do 1990?

(b) Koliki je u svakoj datoj godini procenat troškova po studentu u odnosu na primanja po glavi stanovnika?

(c) Za koliko je od 7 navedenih godina trošak po studentu manji od 1/5 (jedne petine) prihoda po glavi stanovnika?

Rešenje. Niz *a* – prihod po stanovniku; niz *b* – trošak po studentu (dimenzije nizova su *n*=7); promenljiva *g* – godina iz koje su podaci.

$n \leftarrow 7$
učitaj niz *a*
učitaj niz *b*
$g \leftarrow 1930$
$procenat \leftarrow b_1 / a_1 * 100$
    ispiši "Udeo troškova u godini" *g* " je " *procenat*
**ako** *procenat* < 20 **onda**
    $petina \leftarrow 1$
**inače**

[1] Zadatak je preuzet iz zvaničnog priručnika za GRE (Graduate Record Examination) test kvantitaivnog mišljenja.

*petina* ← 0
**kraj ako-onda**
**za** *i* **od** 2 **do** *n*
*g* ← 1930 + (*i* – 1)*10
*rast* ← ($a_i$ - $a_{i-1}$)/10
*procenat* ← $b_i$ / $a_i$ * 100
ispiši "Udeo troškova u godini", *g,* " je ", *procenat*
ispiši "Rast prihoda po godini je ", *rast*
**ako** *procenat* < 20 **onda**
*petina* ← *petina* + 1
**kraj ako-onda**
**do ovde** *i*
*rast* ← ($a_n$ - $a_1$)/((*n*-1)*10)
ispiši "Približan rast po godini je ", *rast*
ispiši "Udeo troškova po studentu manji od jedne petine je u ", *petina*, " dekada"

| *i* | *g* | *procenat* | *rast* | *petina* |
|---|---|---|---|---|
| | 1930 | 10.74 | | 1 |
| 2 | 1940 | 13.65 | 30 | 2 |
| 3 | 1950 | 13.94 | 258 | 3 |
| 4 | 1960 | 15.81 | 324 | 4 |
| 5 | 1970 | 19.84 | 456 | 5 |
| 6 | 1980 | 23.23 | 281 | 5 |
| 7 | 1990 | 24.31 | 408 | 5 |

Približan rast po godini je 293.67
Udeo troškova po studentu manji od jedne petine je u 5 dekada
☺

**Primer 6.** Sastaviti algoritam koji pronalazi vrednost *x* u rastućem nizu $a_1$, $a_2$, $a_3$,..., $a_n$. Algoritam ide redom kroz elemente niza i utvrđuje da li tekući element niza ima vrednost *x*, i daje odgovor '*x* nije pronađeno' ili '*x* je pronađeno'. U drugom slučaju, algoritam određuje i indeks prvog elementa niza koji ima zadatu vrednost.
**Rešenje:**
Neka je $a_1 \le a_2 \le a_3 \le ... \le a_n$ neki niz od *n* elemenata, i neka je *x* vrednost koja se traži u nizu. Pošto je niz rastući, a pretraživanje počinje sa prvim elementom niza, pretraga se završava ako je zadovoljeno nešto od sledećeg:

- Vrednost je pronađena u nizu, ili
- Stiglo se do kraja niza, ili

- Stiglo se do elementa niza koji je veći od traženog elementa.

U poslednja dva slučaja to znači da se element koji se traži ne nalazi u nizu. Indeks pronađenog elementa niza biće vrednost promenljive *k*.

*x* nije pronađeno
$i \leftarrow 1$
**sve dok je** $i \leq n$ **I** $x \geq a_i$ **I** *x* nije pronađeno
    **ako** $a_i = x$ **onda**
        *x* je pronađeno
    **inače**
        $i \leftarrow i + 1$
    **kraj ako-onda**
**do ovde**

Ilustracija funkcionisanja algoritma:
**Ulaz:** $x = 32$

| $a_i$ | 1 | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **indeks** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

Tok rada algoritma:
*x* nije pronađeno
$i \leftarrow 1$

| $i \leq n$ | $x \geq a_i$ | ***x* nije pronađeno** | **Uslov** | $a_i = x$**?** | $i$ | $a_i$ |
|---|---|---|---|---|---|---|
| T | T | T | T | $1 = 32?\perp$ | 2 | 2 |
| T | T | T | T | $2 = 32?\perp$ | 3 | 3 |
| T | T | T | T | $3 = 32?\perp$ | 4 | 5 |
| T | T | T | T | $5 = 32?\perp$ | 5 | 7 |
| T | T | T | T | $7 = 32?\perp$ | 6 | 11 |
| T | T | T | T | $11 = 32?\perp$ | 7 | 13 |
| T | T | T | T | $13 = 32?\perp$ | 8 | 17 |
| T | T | T | T | $17 = 32?\perp$ | 9 | 19 |
| T | T | T | T | $19 = 32?\perp$ | 10 | 23 |
| T | T | T | T | $23 = 32?\perp$ | 11 | 29 |
| T | T | T | T | $29 = 32?\perp$ | 12 | 31 |
| T | T | T | T | $31 = 32?\perp$ | 13 | 37 |
| T | $\perp$ | T | $\perp$ | | | |

*x* nije pronađeno = T
☺

U prethodnom primeru je činjenica da je niz $a_1$, $a_2$, $a_3$,..., $a_n$ uređen korišćena kroz uslov završavanja **while** iskaza. Da bi se **while** iskaz i dalje izvršavao treba tekući element niza da još uvek bude manji od vrednosti koja se traži. Upravo neispunjenost ovog uslova je zaustavila izvršavanje **while** iskaza u gornjem primeru (13-ti element niza je imao vrednost 37 koja je veća od vrednosti 32 koja se tražila).

Međutim, činjenica da je niz uređen (u rastućem ili nekom drugom poretku) može se upotrebiti za mnogo efikasnije utvrđivanje da li neki niz sadrži neku vrednost. Taj algoritam u svakom koraku problem svodi na duplo manji problem. Postupak se zasniva na sledećem: proverava se kako vrednost koja se traži stoji u odnosu na središnji element niza (element čiji je indeks polovina dužine niza). Naravno, ako je središnji element niza upravo element koji se traži, problem je rešen. Ali, ako je središnji element manji od elementa koji se traži, onda su manji i svi elementi ispred njega pa tu polovinu niza treba isključiti iz dalje pretrage. Isto tako, ako je središnji element veći od elementa koji se traži, onda su veći i svi elementi iza njega pa tu polovinu niza treba isključiti iz dalje pretrage. Ovaj postupak se ponavlja za izabranu polovinu niza, onu za koju znamo da mora sadržati vrednost koja se traži (ako je ta vrednost uopšte u nizu).

**Primer 7.** Sastaviti algoritam koji pronalazi vrednost $x$ u rastućem nizu $a_1$, $a_2$, $a_3$,..., $a_n$ korišćenjem algoritma binarnog pretraživanja. Algoritam određuje i indeks elementa niza koji ima zadatu vrednost, ako takva vrednost u nizu postoji ili daje odgovor '$x$ nije pronađeno'.

**Rešenje:**

Neka je $a_1 \le a_2 \le a_3 \le ... \le a_n$ neki niz od $n$ elemenata, i neka je $x$ vrednost koja se u nizu traži. Pretraživanje počinje sa središnjim elementom niza, tj. elementom čiji je indeks $m = (n+1)$ **div** 2. Taj element može da bude jednak traženoj vrednosti, manji od nje ili veći od nje, što u svakom slučaju proizvodi drugačiju akciju:

- $a_m = x$, vrednost je pronađena;
- $a_m < x$, tražena vrednost, ako postoji, je u gornjoj polovini niza, na početku u intervalu $[m+1, n]$;
- $a_m > x$, tražena vrednost, ako postoji, je u donjoj polovini niza, na početku u intervalu $[1, m-1]$.

$x$ nije pronađeno
$poc \leftarrow 1$
$kraj \leftarrow n$
**sve dok je** $poc \le kraj$ **I** $x$ nije pronađeno
    $i \leftarrow (poc + kraj)$ **div** 2
    **ako** $a_i = x$ **onda**

*x* je pronađeno
**inače ako** $a_i < x$ **onda**
*poc* ← *i* + 1
**inače** *kraj* ← *i – 1*
**kraj ako-onda**
**do ovde**

Ilustracija funkcionisanja algoritma (1):
**Ulaz:** $x = 32$

| $a_i$ | 1 | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **indeks** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

Tok rada algoritma:
*x* nije pronađeno
*poc* ← 1
*kraj* ← 14

| *poc* ≤ *kraj* | *x* nije pronađeno | U | i | $a_i$ | $a_i = x$? | $a_i < x$? | *poc* | *kraj* |
|---|---|---|---|---|---|---|---|---|
| T | T | T | 7 | 13 | ⊥ | T | 8 | (14) |
| T | T | T | 11 | 29 | ⊥ | T | 12 | (14) |
| T | T | T | 13 | 37 | ⊥ | ⊥ | (12) | 12 |
| T | T | T | 12 | 31 | ⊥ | T | 13 | (12) |
| ⊥ | T | ⊥ | | | | | | |

Algoritam završava sa radom sa rezultatom '*x* nije pronađeno'

Ilustracija funkcionisanja algoritma (2):
**Ulaz:** $x = 11$

| $a_i$ | 1 | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **indeks** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

Tok rada algoritma:
*x* nije pronađeno
*poc* ← 1
*kraj* ← 14

| *poc* ≤ *kraj* | *x* nije pronađeno | U | i | $a_i$ | $a_i = x$? | $a_i < x$? | *poc* | *kraj* |
|---|---|---|---|---|---|---|---|---|
| T | T | T | 7 | 13 | ⊥ | ⊥ | (1) | 6 |
| T | T | T | 3 | 3 | ⊥ | T | 4 | (6) |
| T | T | T | 5 | 7 | ⊥ | T | 6 | (6) |
| T | T | T | 6 | 11 | T | | (6) | (6) |
| T | ⊥ | ⊥ | | | | | | |

Algoritam završava sa radom sa rezultatom '$x$ je pronađeno' i $i = 6$
☺

U prethodnim primerima 6 i 7 smo pretpostavljali da radimo sa nizom koji je uređen. Sada ćemo predstaviti dva algoritma koji od neuređenog niza proizvode, recimo, rastući uređeni niz. Ovo su dva najjednostavnija i samim tim najneefikasnija algoritma.

Prvi algoritam se naziva *uređivanje niza izborom najmanjeg* (*selection sort*). Ovim algoritmom se u svakom koraku redom na sledeći element niza postavlja najmanja vrednosti od svih preostalih vrednosti. To znači da se za svaki element niza mora redom proći kroz sve preostale elemente niza da bi se utvrdilo koji od njih je najmanji.

**Primer 8.** Sastaviti algoritam koji niz $a_1$, $a_2$, $a_3$,..., $a_n$ uređuje metodom izbora najmanjeg elementa u uređeni niz $a_1 \le a_2 \le a_3 \le ... \le a_n$.

**za** $i$ **od** 1 **do** $n$-1
    **za** $j$ **od** $i$ + 1 **do** n
        **ako** $a_i > a_j$ **onda**
            $a_i \leftrightarrow a_j$
        **kraj ako-onda**
    **do ovde** $j$
**do ovde** $i$

Ilustracija funkcionisanja algoritma:
**Ulaz**: $n = 6$

| $a_i$ | 3 | 7 | 2 | 5 | 1 | 4 |
|---|---|---|---|---|---|---|
| Indeks | 1 | 2 | 3 | 4 | 5 | 6 |

| ***i*** | ***j*** | $a_i > a_j$ | $a_i \leftrightarrow a_j$ | $a_1, a_2, a_3, ..., a_n$ |
|---|---|---|---|---|
| 1 | 2 | 3 > 7 NE | | |
| | 3 | 3 > 2 DA | $a_1 \leftarrow 2, a_3 \leftarrow 3$ | 2 7 3 5 1 4 |
| | 4 | 2 > 5 NE | | |
| | 5 | 2 > 1 DA | $a_1 \leftarrow 1, a_5 \leftarrow 2$ | **1** 7 3 5 2 4 |
| | 6 | 1 > 4 NE | | |
| 2 | 3 | 7 > 3 DA | $a_2 \leftarrow 3, a_3 \leftarrow 7$ | 1 3 7 5 2 4 |
| | 4 | 3 > 5 NE | | |
| | 5 | 3 > 2 DA | $a_2 \leftarrow 2, a_5 \leftarrow 3$ | **1 2** 7 5 3 4 |

| | | | | |
|---|---|---|---|---|
| | 6 | 2 > 4 NE | | |
| 3 | 4 | 7 > 5 DA | $a_3 \leftarrow 5, a_4 \leftarrow 7$ | 1 2 5 7 3 4 |
| | 5 | 5 > 3 DA | $a_3 \leftarrow 3, a_5 \leftarrow 5$ | **1 2 3** 7 5 4 |
| | 6 | 3 > 4 NE | | |
| 4 | 5 | 7 > 5 DA | $a_4 \leftarrow 5, a_5 \leftarrow 7$ | 1 2 3 5 7 4 |
| | 6 | 5 > 4 DA | $a_4 \leftarrow 4, a_6 \leftarrow 5$ | **1 2 3 4** 7 5 |
| 5 | 6 | 7 > 5 DA | $a_5 \leftarrow 5, a_6 \leftarrow 7$ | **1 2 3 4 5 7** |

☺

Drugi algoritam se naziva *uređivanje razmenom susednih elemenata* (ili *mehurići*, prema engleskom *bubble sort*). Ovaj algoritam se sastoji u tome da se u svakom koraku prolazi kroz sve elemente niza (osim onih s kraja niza koji su prethodnim koracima došli na svoje mesto) i pri tom se porede dva susedna elementa, koji pri tom razmenjuju mesta ako nisu u željenom redosledu. Takvim postupanjem se u svakom koraku najveći element smešta na poslednju poziciju obrađivanog dela niza.

**Primer 9.** Sastaviti algoritam koji niz $a_1$, $a_2$, $a_3$,..., $a_n$ uređuje metodom razmene susednih elementa u uređeni niz $a_1 \le a_2 \le a_3 \le ... \le a_n$.

**za** *i* **od** *n*-1 **do** 1 **po koraku** -1
    **za** *j* **od** 1 **do** i
        **ako** $a_j > a_{j+1}$ **onda**
            $a_j \leftrightarrow a_{j+1}$
        **kraj ako-onda**
    **do ovde** *j*
**do ovde** *i*

Ilustracija funkcionisanja algoritma:
**Ulaz**: $n = 6$

| $a_i$ | 3 | 7 | 2 | 5 | 1 | 4 |
|---|---|---|---|---|---|---|
| Indeks | 1 | 2 | 3 | 4 | 5 | 6 |

| ***i*** | ***j*** | $a_j > a_{j+1}$ | $a_j \leftrightarrow a_{j+1}$ | $a_1, a_2, a_3, ..., a_n$ |
|---|---|---|---|---|
| 5 | 1 | 3 > 7 NE | | |
| | 2 | 7 > 2 DA | $a_2 \leftarrow 2, a_3 \leftarrow 7$ | 3 2 7 5 1 4 |
| | 3 | 7 > 5 DA | $a_3 \leftarrow 5, a_4 \leftarrow 7$ | 3 2 5 7 1 4 |

| | | | | |
|---|---|---|---|---|
| | 4 | 7 > 1 DA | $a_4 \leftarrow 1, a_5 \leftarrow 7$ | 3 2 5 1 7 4 |
| | 5 | 7 > 4 DA | $a_5 \leftarrow 4, a_6 \leftarrow 7$ | 3 2 5 1 4 **7** |
| 4 | 1 | 3 > 2 DA | $a_1 \leftarrow 2, a_2 \leftarrow 3$ | 2 3 5 1 4 7 |
| | 2 | 3 > 5 NE | | |
| | 3 | 5 > 1 DA | $a_3 \leftarrow 1, a_4 \leftarrow 5$ | 2 3 1 5 4 7 |
| | 4 | 5 > 4 DA | $a_4 \leftarrow 4, a_5 \leftarrow 5$ | 2 3 1 4 **5 7** |
| 3 | 1 | 2 > 3 NE | | |
| | 2 | 3 > 1 DA | $a_2 \leftarrow 1, a_3 \leftarrow 3$ | 2 1 3 **4 5 7** |
| | 3 | 3 > 4 NE | | |
| 2 | 1 | 2 > 1 DA | $a_1 \leftarrow 1, a_2 \leftarrow 2$ | 1 2 **3 4 5 7** |
| | 2 | 2 > 3 NE | | |
| 1 | 1 | 1 > 2 NE | | **1 2 3 4 5 7** |

☺

**Primer 10.** Neka se u nizu $a_1$, $a_2$, $a_3$,..., $a_n$ nalaze ocene položenih predmeta jednog studenta. Odrediti vrednosti niza $b_1$, $b_2$, $b_3$, $b_4$, $b_5$ koje sadrže koliko student ima kojih pozitivnih ocena ($b_1$ - broj šestica, $b_5$ – broj desetki), a zatim odrediti njegovu prosečnu ocenu.

U rešenju ovog zadatka će se ocena studenta koristiti za izračunavanje indeksa niza *b* koji služi za prebrojavanje ocena: u $b_1$ će se smeštati broj šestica (6-5=1), u $b_2$ broj sedmica (7-5=2), itd.

```
za i od 1 do 5        /* na početku brojači svih ocena su 0 */
      b1 ← 0
do ovde i
za j od 1 do n        /* prolazi se kroz sve ocene studenta */
      baj-5 ← baj-5 + 1
do ovde j
socena ← 0            /* računanje srednje ocene */
za i od 1 do 5
      socena ← socena + bi * (i + 5)
do ovde i
socena ← socena/n
```

Ilustracija funkcionisanja algoritma:
**Ulaz**: $n = 6$

| $a_i$ | 6 | 8 | 10 | 7 | 10 | 9 |
|---|---|---|---|---|---|---|
| Indeks | 1 | 2 | 3 | 4 | 5 | 6 |

| ***j*** | $a_j$ | $a_j$**-5** | $b_1, b_2, b_3, b_4, b_5$ |
|---|---|---|---|
| | | | 0, 0, 0, 0, 0 |
| 1 | 6 | 1 | 1, 0, 0, 0, 0 |
| 2 | 8 | 3 | 1, 0, 1, 0, 0 |
| 3 | 10 | 5 | 1, 0, 1, 0, 1 |
| 4 | 7 | 2 | 1, 1, 1, 0, 1 |
| 5 | 10 | 5 | 1, 1, 1, 0, 2 |
| 6 | 9 | 4 | 1, 1, 1, 1, 2 |

| ***i*** | $b_i$ | ***i*+5** | ***socena*** |
|---|---|---|---|
| | | | 0 |
| 1 | 1 | 6 | 6 |
| 2 | 1 | 7 | 13 |
| 3 | 1 | 8 | 21 |
| 4 | 1 | 9 | 30 |
| 5 | 2 | 10 | 50 |
| | | | 8.33 |

☺

**Zadatak 2.** Sledećom tabelom su dati podaci o broju stanovnika u 5 gradova u SAD-u aprila 2000. godine:

| **grad** | **broj stanovnika** |
|---|---|
| Los Anđeles | 3.695.000 |
| Njujork | 8.008.000 |
| Hjuston | 1.954.000 |
| Filadelfija | 1.518.000 |
| Čikago | 2.896.000 |

Ukupan broj stanovnika u SAD aprila 2000. godine je bio 281.422.000. Koliki procenat ukupnog broja stanovnika SAD-a čini broj stanovnika u tri najveća grada? Sastaviti program koji rešava zadatak i za slučaj da je lista gradova SAD-a mnogo duža (proizvoljna).

*grad* – niz koji sadrži broj stanovnika u gradovima SAD
*n* – broj gradova u listi
*SAD* – broj stanovnika u SAD
*max1* – broju stanovnika u gradu sa najvećim brojem stanovnika
*max2* – broj stanovnika u gradu drugom po redu
*max3* – broj stanovnika u gradu trećem po redu

*max1* = 0
*max2* = 0
*max3* = 0
**za** *i* **od** 1 **do** *n*
    **ako** $grad_i > max1$ **onda**
        *max3* = *max2*
        *max2* = *max1*
        *max1* = $grad_i$
    **inače ako** $grad_i > max2$ **onda**
        *max3* = *max2*
        *max2* = $grad_i$
    **inače ako** $grad_i > max3$ **onda**
        *max3* = $grad_i$
    **kraj ako onda**
**do ovde** *i*

*procenat* = (*max1* + *max2* + *max3*) * 100 / *SAD*

Ilustracija funkcionisanja algoritma:
*n* = 5
*SAD* = 281422000

| *i* | $grad_i$ | $grad_i > max1$ | $grad_i > max2$ | $grad_i > max3$ | *max1* | *max2* | *max3* |
|---|---|---|---|---|---|---|---|
| | | | | | 0 | 0 | 0 |
| 1 | 3695 | T | | | 3695 | 0 | 0 |
| 2 | 8008 | T | | | 8008 | 3695 | 0 |
| 3 | 1954 | F | F | T | 8008 | 3695 | 1954 |
| 4 | 1518 | F | F | F | 8008 | 3695 | 1954 |
| 5 | 2896 | F | F | T | 8008 | 3695 | 2896 |

*procenat* = (8008000+3695000+2896000)/28142000*100 = 5.19%
☺

U programskim jezicima se nizovi realizuju tako što se elementi niza koji su svi istog tipa skladište u susedne memorijske lokacije. Na taj način je omogućen direktan pristup svakom elementu niza jer se adresa svakog elementa niza može lako izračunati na osnovu adrese prvog elementa niza i memorijskog zauzeća svakog elementa niza po formuli:

adresa_elementa_Ai = adresa_elementa_A1 + (i-1)*broj_bajtova_za_element

Nizovi se u programskom jeziku VBA moraju deklarisati tako što se navodi kog je tipa svaki element niza i koliko elementa najviše ima niz. Niz može da bude statički i onda se maksimalna dimenzija niza zadaje prilikom deklaracije niza:

```
    Const MAXDIM As Integer = 20 'staticki niz
    Dim niz(MAXDIM) As Integer ' stvarna dimenzija n mora biti < MAXDIM
```

Ako je niz dinamički on se onda deklariše bez navođenja maksimalne dimenzije niza:

```
    Dim niz() As Integer 'dinamicki niz; dimenziju zadaje korisnik
```

Kada je poznata maksimalna (ili stvarna) dimenzija niza mora se odvojiti memorijski prostor za taj niz, pre nego što se bilo šta drugo s nizom radi.

```
    'Odvajanje memorijskog prostora za niz
    ReDim niz(n) 'promena dimenzije niza (ReDimension)
```

Ako su nizovi parametri funkcija ili procedura, oni se uvek prenose po referenci, tj. u pozvanoj proceduri je niz samo drugo ime za memorijske lokacije koje se koriste u proceduri koja ga je pozvala.

**Primer 11.** Sastaviti VBA funkciju koja pronalazi najveći element niza koji je argument te funkcije (kao i njegova dimenzija, tj. broj elemenata).

```
Function maxNiz(ByRef niz() As Integer, _
     ByVal n As Integer) As Integer
    Dim i As Integer 'brojac
'pomocna promenljiva koja čuva najveću tekuću vrednost
    Dim pom As String
    pom = niz(1)
    For i = 2 To n
        If niz(i) > pom Then
           pom = niz(i)
        End If
```

```
    Next i
    maxNiz = pom
End Function
```

Ilustracija funkcionisanja algoritma:
**Ulaz**: $n = 6$

| $a_i$ | 3 | 6 | 2 | 5 | 7 | 4 |
|---|---|---|---|---|---|---|
| Indeks | 1 | 2 | 3 | 4 | 5 | 6 |

| ***i*** | $a_i$ | $a_i >$ ***pom*** | ***pom*** |
|---|---|---|---|
| | | | 3 |
| 2 | 6 | T | 6 |
| 3 | 2 | ⊥ | 6 |
| 4 | 5 | ⊥ | 6 |
| 5 | 7 | T | 7 |
| 6 | 4 | ⊥ | 7 |

*maxNiz* = 7

**Primer 12.** Sastaviti VBA program koji niz reči (niski) koji se učitava s ulaza sortira u rastućem redosledu (kolacione sekvence) koriteći algoritam uređivanja izborom najmanjeg.

```
Sub testRadSaNizovima()
    'Deklaracija promenljivih
    Dim n As Integer     'Dimenzija niza
    'dinamicki niz; dimenziju zadaje korisnik
    Dim niz() As String
    Dim sniz As String   'Zapis niza u vidu niske
    unosNiza niz, n      'Ulaz
    sniz = zapisNiza(niz, n)    'Ispisivanje niza
    If sniz = "" Then
        MsgBox "Niz je prazan"
        Exit Sub
    Else
        MsgBox "Polazni niz: " & sniz
    End If
    selectionSort niz, n 'Sortiranje niza
    'Ispisivanje niza, ponovo jer se niz promenio
    sniz = zapisNiza(niz, n)
    MsgBox "Sortirani niz: " & vbCrLf & sniz
End Sub

Sub selectionSort(ByRef niz() As String, _
     ByVal n As Integer)
```

```
    Dim i As Integer, j As Integer 'brojaci
'pomocna promenljiva za razmenu sadrzaja promenljivih
    Dim p As String
    For i = 1 To n - 1
        For j = i + 1 To n
            If niz(j) < niz(i) Then
                p = niz(i)
                niz(i) = niz(j)
                niz(j) = p
            End If
        Next j
    Next i
End Sub

Function zapisNiza(ByRef niz() As String, _
     ByVal n As Integer) As String
    Dim i As Integer
    If n <= 0 Then
        zapisNiza = ""
    Else
        zapisNiza = niz(1)
        For i = 2 To n
            zapisNiza = zapisNiza & " " & niz(i)
        Next i
    End If
End Function

Sub unosNiza(ByRef niz() As String, ByRef n As Integer)
    n = InputBox("Unesite broj elemenata niza:")
    'Odvajanje memorijskog prostora za niz
    ReDim niz(n) 'promena dimenzije niza (ReDimension)
    'Inicijalizacija niza
    nizSaTastature niz, n
End Sub

Sub nizSaTastature(ByRef niz() As String, ByVal n As Integer)
    Dim i As Integer
    For i = 1 To n
        niz(i) = InputBox("Element niz(" & i & ") =")
    Next i
End Sub
```

Ako se sa ulaza (tastature) učita niz od 10 reči: mama, tata, brat, strina, ujak, ujna, sestra, stric, baka, deka, program će ispisati:

Sortirani niz:

baka, brat, deka, mama, sestra, stric, strina, tata, ujak, ujna
☺