

Osnovi programiranja



2024/25



Nizovi



Nizovi

- Standardni i nestandardni prosti tipovi podataka (celobrojni, realni, logički, znakovni, nabrojivi) mogu biti sasvim dovoljni pri rešavanju manjih i jednostavnijih problema.
- Međutim, u slučaju rešavanja složenijih problema često je potrebno baratati velikim brojem podataka. Veliki broj podataka bi zahtevao postojanje isto tolikog broja promenljivih u koje bi se mogle smestiti njihove vrednosti.
- Na primer obračun plata u nekom preduzeću.

```
float pera,mika,laza,...,zoran1,zoran2,...,pera1254;  
pera = pera_sati * cena_rada;  
mika = mika_sati * cena_rada;  
laza = laza_sati * cena_rada;  
  
...  
zoran1 = zoran1_sati * cena_rada;  
zoran2 = zoran2_sati * cena_rada;  
  
...  
pera1254 = pera1254_sati * cena_rada;
```

Nizovi

- Rešenje ovakvih problema u programskom jeziku C je moguće korišćenjem **nizovnog tipa podataka**, odnosno **nizova**.
- Niz je ograničen uređen skup promenljivih istog tipa, koje se nazivaju **elementima** ili **članovima** niza. Svakom elementu niza pridružen je indeks, uz pomoć koga je moguće pristupiti odgovarajućem elementu.

a

4	8	3	11	6	5	5	17	13	6
---	---	---	----	---	---	---	----	----	---

Nizovi

- Da bi bilo moguće definisati niz, neophodno je znati interval u kome se kreću indeksi niza, kao i tip elemenata niza.
- Nizovni tip se u programskom jeziku C definiše tako što se navede tip elemenata pa naziv niza iza kojeg u uglastim zagradama sledi dužina niza (broj elemenata niza).
- U programskom jeziku C indeksi niza uvek počinju od 0, pa je indeks poslednjeg elementa u nizu za jedan manji od broja elemenata. Ovakva definicija nizovnog tipa se može koristiti za definisanje novog tipa u okviru naredbe **typedef** ili za deklarisanje nizovnih promenljivih.

```
typedef <tip_elementata> <novi_tip>[<dužina_niza>];
```

ili

```
<tip_elementata> <naziv_promenljive>[<dužina_niza>];
```

Nizovi

a	4	8	3	11	6	5	5	17	13	6
	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

- Za pristupanje određenom elementu niza, pored konstanti se mogu koristiti i promenljive i izrazi, ali pri tome treba voditi računa da njihov tip mora biti celobrojan i unutar definisanog intervala niza:
- `p=2; /* celobrojna promenljiva dobija vrednost 2 */`
- `a[p+3]=7; /* ispravno, jer je indeks 5 unutar opsega 0..9 */`
- `a[p+8]=3; /* neispravno, jer indeks 10 izlazi iz opsega 0..9 */`
- `a[1+a[p+3]]=a[2]; /* ispravno */`

Primer 1

- Napisati program u kom korisnik unosi broj radnih sati s_i koje je ostvario svaki od n radnika, cenu radnog sata c , a zatim se izračunava plata svakog radnika po obrascu $p_i = s_i * c$.

```
#include <stdio.h>
main()
{
    int i, n;
    float s[100], p[100];
    float c;
    printf("Unesite broj radnika:");
    scanf("%d", &n);
```

Primer 1

```
//Unos broja radnih sati za svakog radnika
for(int i = 0; i < n; i++)
{
    printf("Unesite broj radnih sati %d. radnika:\n", i+1);
    scanf("%f", &s[i]);
}
printf("Unesite cenu radnog sata:\n");
scanf("%f", &c);
//Racunanje i stampanje plata
for(int i = 0; i < n; i++)
{
    p[i] = s[i] * c;
    printf("Plata %d. radnika je %.2f dinara.\n", i+1, p[i]);
}
}
```

Primer 2

- Napisati program koji učitava dva realna niza istih dužina, a zatim pozivanjem potprograma izračunava njihov zbir i skalarni proizvod.
- Zbir dva niza (vektora) a i b dužine n je takođe niz dužine n , pri čemu su članovi novog niza s definisani kao: $s_i = a_i + b_i$
- Skalarni proizvod dva niza (vektora) a i b dužine n je skalar p definisan kao: $p = a_1 \cdot b_1 + a_2 \cdot b_2 + \dots + a_n \cdot b_n = \sum_{i=1}^n a_i \cdot b_i$

```
#include <stdio.h>
typedef float niz[100];

void zbir(niz a, niz b, niz c, int n)
{
    int i;
    for(i = 0; i < n; i++)
        c[i] = a[i] + b[i];
}
```

Primer 2

```
float skalarni(niz a, niz b, int n)
{
    int i;
    float suma = 0.0;
    for(i = 0; i < n; i++)
        suma += a[i] * b[i];
    return suma;
}
main()
{
    int n, i;
    niz a, b, s;
    float p;
    printf("Unesite broj elemenata niza:\n");
    scanf("%d", &n);
    printf("Unesite elemente prvog niza:\n");
    for(i = 0; i < n; i++)
        scanf("%f", &a[i]);
```

Primer 2

```
printf("Unesite elemente drugog niza:\n");
    for(i = 0; i < n; i++)
scanf("%f", &b[i]);
zbir(a, b, s, n);

p = skalarni(a, b, n);

printf("Zbir vektora je:\n");
for(i = 0; i < n; i++)
    printf("%10.4f", s[i]);

printf("\n\n");
printf("Skalarni proizvod vektora je: %10.4f", p);
}
```

Pretraživanje nizova

- Pretraživanje je proces koji za cilj ima pronalaženje elementa niza, koji zadovoljava unapred definisani kriterijum. Najčešće je to traženje elementa koji sadrži određeni podatak, koji nazivamo **ključ**. U zavisnosti od tipa strukture podataka u kojoj se vrši pretraživanje, algoritmi mogu biti manje ili više kompleksni.
- **Sekvencijalno pretraživanje** je sigurno najjednostavniji algoritam za pretraživanje nizova. Ovaj algoritam se naziva još i **linearno pretraživanje** iz razloga što se traženje određenog elementa vrši tako što se ispituje jedan po jedan element niza sve dok se ne nađe element koji zadovoljava zadate kriterijume ili se ne dođe do kraja niza.

Primer 3

- Posmatrajmo niz celih brojeva a dužine n . Napisati program koji nalazi poziciju prvog pojavljivanja zadatog celog broja t u nizu a . Ukoliko niz ne sadrži traženi element, funkcija treba da vrati -1.

```
#include <stdio.h>
typedef int niz[100];
int trazi(niz a, int n, int t)
{
    int i;
    int indeks = -1;
    for(i = 0; i < n; i++)
        if(a[i] == t)
        {
            indeks = i;
            break;
        }
    return indeks;
}
```

Primer 3

```
main()
```

```
{
```

```
    int i, broj, trazen_i, indeks;
```

```
    niz a;
```

```
    printf("Unesite broj elemenata niza:\n");
```

```
    scanf("%d", &broj);
```

```
    printf("Unesite elemente niza:\n");
```

```
    for(i = 0; i < broj; i++)
```

```
    {
```

```
        printf("Unesite %d. element niza:\n", i+1);
```

```
        scanf("%d", &a[i]);
```

```
    }
```

```
    printf("Unesite element koji trazite:\n");
```

```
    scanf("%d", &trazen_i);
```

```
    indeks = trazi(a, broj, trazen_i);
```

```
    if(indeks > - 1)
```

```
        printf("Trazeni element se nalazi na poziciji %d.\n", indeks);
```

```
    else
```

```
        printf("Trazeni element ne postoji.\n");
```

```
}
```