

Računske vježbe 3

Programiranje I

1. Napisati program kojim se učitava niz cijelih brojeva X , dužine N , a zatim i pozitivan jednocifren broj P . Program treba da formira novi niz Y koji se sastoji samo od elemenata niza X čija je cifra jedinica jednaka broju P . Na izlazu stampati niz Y . U slučaju da P nije pozitivan jednocifren broj, prikazati odgovarajuću poruku.

Na samom početku deklariramo potrebne promjenljive. Pored onih definisanih u tekstu zadatka biće nam potrebna jedna za brojač petlji, ali i jedna koja će nam voditi računa o tome koliko smo elemenata x upisali u y :

```
int i, n, p, k = 0, x[20], y[20];
```

gdje smo sa $x[20]$ i $y[20]$ deklarirali nizove dužine 20, ali nismo u njih unijeli ništa. Memorija je zauzeta pa će elementi imati neku vrijednost koja nam nije od koristi. Nakon toga vršimo unos broja članova niza, a onda zatim unos cijelog broja p . Zatim provjeravamo validnost unosa cijelog broja p :

```
if(p <= 0) printf("Nije unijet pozitivan cijeli broj!");
```

gdje korisnika obavještavamo da je unos pogrešan. Ukoliko je unos ispravan, izvršiće se ono što se nalazi u `else` grani. Nakon toga vršimo unos n cijelih brojeva, smještamo ih u niz x , a neposredno nakon smještanja provjeravamo uslov postavljen zadatkom i ukoliko je zadovoljen vršimo upis u niz y :

```
printf("Unesite članove niza:\n");
for(i = 0; i < n; i++)
{
    scanf("%d", x + i); // moze i scanf("%d", &x[i]);
    if(x[i]%10 == p)
        y[k++] = x[i];
}
```

gdje vidimo da adresu člana niza možemo dobiti sa $x+i$ ili sa $\&x[i]$. U prvom slučaju, x predstavlja adresu prvog elementa, a $x+i$ adresu i -tog elementa. Uočite na koji smo način inkrementirali k , što je fina pogodnost sintakse programskog jezika C. Nakon toga šampamo rezultujući niz, pa je konačno rješenje:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i, n, p, k = 0, x[20], y[20];
6
7     printf("Unesite broj članova niza:\n");
8     scanf("%d", &n);
9
10    printf("Unesite cijeli broj:\n");
11    scanf("%d", &p);
12
13    if(p <= 0) printf("Nije unijet pozitivan cijeli broj!");
14    else
15    {
16        printf("Unesite članove niza:\n");
```

```

17     for(i = 0; i < n; i++)
18     {
19         scanf("%d", x + i); // moze i scanf("%d", &x[i]);
20         if(x[i]%10 == p)
21             y[k++] = x[i];
22     }
23
24     if(k > 0)
25     {
26         printf("Rezultujuci niz je:\n");
27         for(i = 0; i < k; i++)
28             printf("%d ", y[i]);
29     }
30     else
31         printf("Rezultujuci niz je prazan.");
32 }
33 }

```

Mogli smo koristiti i skraćeni oblik:

```
if(k)
```

zato što programski jezik C svaku nenultu vrijednost tretira kao logičko TRUE dok je 0 ekvivalentna logičkom FALSE.

2. Napisati program kojim se definiše niz cijelih brojeva **X** i vrši sumiranje elemenata tog niza korišćenjem pomoćne pokazivačke promjenljive. Prilagoditi program tako da eventualna izmjena broja elemenata niza, u njegovoj inicijalizaciji, ne zahtijeva intervenciju na programu.

Kako u ovom slučaju korisnik ne unosi dužinu niza kao ulazni podatak, potrebno ju je izračunati na osnovu definisanog niza. Pomoću operatora `sizeof()` moguće je dobiti veličinu operanda. Veličina predstavlja količinu memorije koju operand zauzima u bajtima. **Pažnja**, veličina i dužina nisu isto. Dužina predstavlja broj elemenata niza i može se dobiti na sljedeći način:

```
int length = sizeof(x) / sizeof(x[0]);
```

tako što znamo da je ukupna količina memorije koju niz zauzima jednaka zbiru količina memorije koju zauzimaju njegovi elementi. Rješenje zadatka je:

```

1 #include <stdio.h>
2
3 int main()
4 {
5     int x[] = {1, 2, 6, -2, 7};
6     int *p, i, sum = 0;
7     int length = sizeof(x) / sizeof(x[0]);
8     for(i = 0; i < length; i++)
9     {
10         p = &x[i];
11         sum += *p;
12     }
13     printf("Dobijena suma je: %d\n", sum);
14 }

```

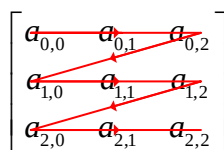
Uočite da `*` predstavlja operator dereferenciranja, tj. dobijanja sadržaja na određenoj memorijskoj adresi. Suprotan je od operatora referenciranja `&`.

3. Napisati program kojim se učitava matrica cijelih brojeva A , dimenzija $N \times M$, i kojim se unijeta matrica štampa prateći strelice prikazane na slici. Na primjer, za matricu:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \end{bmatrix}$$

redosljed štampanja treba da bude: 1, 11, 21, 31, 32, 22, 12, 2, 3, 13, 23, 33, 34, 24, 14, 4.

U programskom jeziku C matrica predstavlja niz nizova koji se u memoriji čuvaju u poretku po vrstama (engl. *row-major order*), ali mi elementima možemo pristupiti onako kako mi to želimo, sve dok ne izađemo iz memorijskog opsega koji ta matrica zauzima. Kako se u C-u elementi indeksiraju od 0, prvom elementu matrice A pristupamo sa $A[0][0]$, a posljednjem sa $A[N-1][M-1]$, gdje N i M predstavljaju broj vrsta, odnosno kolona. Na slici 1 prikazani su indeksi proizvoljne matrice A , dimenzija 3×3 , i poredak po vrstama.



Slika 1: Indeksi elemenata matrice A i poredak po vrstama.

U našem slučaju, potrebno je elemente parnih kolona izlistati od početnog do krajnjeg indeksa vrste, a neparne obratno. Konačno, rješenje zadatka je:

```

1 #include <stdio.h>
2
3 int main()
4 {
5     int i, j, n, m, a[20][20];
6
7     printf("Unesite dimenzije matrice:\n");
8     scanf("%d %d", &n, &m);
9
10    printf("Unesite elemente matrice:\n");
11
12    for(i = 0; i < n; i++)
13        for(j = 0; j < m; j++)
14            scanf("%d", &a[i][j]);
15
16    for(j = 0; j < m; j++)
17        if(j%2 == 0)
18            for(i=0; i<n; i++)
19                printf("%d ", a[i][j]);
20        else
21            for(i = n - 1; i >= 0; i--)
22                printf("%d ", a[i][j]);
23 }
```

Velike zagrade nisu korištene za spoljašnju for petlju zato što se `if-else` tretira KAO JEDNA NAREDBA, pa nije obavezno koristiti zagrade.