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 štampati niz Y. U slučaju da P nije pozitivan jednocifren broj, prikazati odgovarajuću poruku.

Na samom početku deklarišemo 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 $y\,[20]$ deklarisali 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 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 clanove 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 štampamo rezultujući niz, pa je konačno rješenje:

```
#include <stdio.h>

int main()
{
    int i, n, p, k = 0, x[20], y[20];

    printf("Unesite broj clanova niza:\n");
    scanf("%d", &n);

    printf("Unesite cijeli broj:\n");
    scanf("%d", &p);

    if(p <= 0) printf("Nije unijet pozitivan cijeli broj!");
    else
    {
        printf("Unesite clanove 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];
        }

        if(k > 0)
        {
            printf("Rezultujuci niz je:\n");
            for(i = 0; i < k; i++)
                printf("%d ", y[i]);
        }
        else
            printf("Rezultujuci niz je prazan.");
    }
}
```

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:

```
#include <stdio.h>

int main()
{
    int x[] = {1, 2, 6, -2, 7};
    int *p, i, sum = 0;
    int length = sizeof(x) / sizeof(x[0]);
    for(i = 0; i < length; i++)
    {
        p = &x[i];
        sum += *p;
    }
    printf("Dobijena suma je: %d\n", sum);
}
```

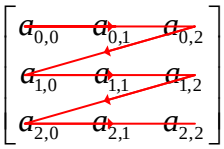
Uočite da * predstavlja operator dereferenciranja, tj. dobijanja sadržaja na određenoj memorijskoj adresi. Suprotan je od operatora referenciranja &.

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 pristupati 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 poslednjem sa A[N-1][M-1], gdje N i M predstavljaju broj vrsta, odnosno kolona. Na slici 1 prikazani su indeksi 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:

```c
#include <stdio.h>

int main()
{
    int i, j, n, m, a[20][20];

    printf("Unesite dimenzije matrice:\n");
    scanf("%d %d", &n, &m);

    printf("Unesite elemente matrice:\n");

    for(i = 0; i < n; i++)
        for(j = 0; j < m; j++)
            scanf("%d", &a[i][j]);

    for(j = 0; j < m; j++)
        if(j%2 == 0)
            for(i=0; i<n; i++)
                printf("%d ", a[i][j]);
        else
            for(i = n - 1; i >= 0; i--)
                printf("%d ", a[i][j]);
}
```

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