Računske vježbe 2

Programiranje I

1. Napisati program koji za unijeti prirodni broj računa zbir njegovih cifara, a nakon toga ga štampa.

C program počinje dijelom koji se naziva pretprocesor i počinje sa #. U našem slučaju mi ćemo ga koristiti za uključivanje zaglavlja (engl. *header*). Zaglavlje je ništa drugo do fajl sa ekstenzijom **.h** u kojem su deklarisane funkcije, konstante i makroi koje projekti dijele. Nazivu zaglavlja koje se uključuje prethodi ključna riječ **include**, a sam naziv zaglavlja se definiše unutar <>. Mi ćemo za početak uključivati samo jedno zaglavlje:

```
#include <stdio.h>
```

odnosno **St**andar**d** **I**nput **O**utput - **stdio** koje nam omogućava da koristimo funkcije kao što su *printf()* i *scanf()*. Kod pišemo unutar funkcije *main()* zato što operativni sistem poziva ovu funkciju prilikom pokretanja programa. Sa funkcijama ćemo se upoznati na nekom od narednih časova. Na početku funkcije definišemo promjenljive koje ćemo koristiti unutar nje. Ovaj dio naredbi se obično piše na početku radi preglednosti, ali to nije obavezujuće. Ono što jeste obavezujuće jeste da se promjenljiva mora deklarisati prije prvog korišćenja u programu. Potrebno je izračunati zbir cifara unijetog prirodnog broja. Drugim riječima, potrebne su nam dvije promjenljive:

```
int sum = 0, num;
```

od kojih **num** nije neophodno inicijalizovati jer će vrijednost unijeti korisnik, dok je **sum** neophodno postaviti na 0 jer će u suprotnom rezultat biti netačan! Obratite pažnju da su promjenljive definisane na engleskom što je programerska praksa i čega ćemo se pridržavati na ovim vježbama, ali u provjerama znanja nije obavezujuće. Sada je potrebno od korisnika zatražiti da unese broj, a onda mu unos i omogućiti. Pošto naši programi nemaju grafički korisnički interfejs, sa korisnikom komuniciraju putem konzole. To ćemo uraditi pomoću:

```
printf("Unesite prirodan broj:\n");
scanf("%d", &num);
```

gdje prvom linijom korisniku prikazujemo poruku u konzolnom prozoru, a drugom učitavamo uneseni broj. Uočiti argumente funkcije *scanf()*. Prvim argumentom definišemo tip promjenljive (u ovom slučaju se radi o cjelobrojnoj vrijednosti pa pišemo **%d**), a drugim prosljeđujemo adresu promjenljive na koju će se vrijednost upisati. To nam omogućava adresni operator **&**. Kada bismo ga izostavili, funkciji bi se proslijedila vrijednost promjenljive, a ne njena adresa. Međutim, funkcija bi tu vrijednost interpretirala kao adresu što nema smisla i doći će do greške.

Sada se možemo vratiti na zadatak. Poslednju cifru cijelog broja dobijamo kao ostatak dijeljenja toga broja i broja 10. Nakon dobijanja cifre, broj umanjujemo 10 puta odnosno ,,brišemo" poslednju cifru i nastavljamo dalje. Kako ne znamo koliko cifri broj ima, zaustavljamo se kada broj svedemo na 0. Pošto ne znamo koliko će se operacija izvršiti koristićemo **while** ciklus. Odgovarajući while ciklus bi bio:

```
while (num != 0)
{
    sum += num % 10;
    num /= 10;
}
```

a od nas se očekuje i da odštampamo rezultat:

```
printf("Suma cifara prirodnog broja je: %d\n", sum);
```

gdje pomoću čuvara mjesta (engl. *placeholder*) *%d* odnosno specifikatora formata (engl. *format specifier*) funkciji *printf()* naglašavamo da na tom mjestu očekuje cjelobrojnu vrijednost. Uočite kako sada prosljeđujemo vrijednost *sum*, a ne njenu adresu. Zašto? Konačno, rješenje zadatka je:

```
#include <stdio.h>

int main()
{
    int sum = 0, num;
    printf("Unesite prirodan broj:\n");
    scanf("%d", &num);
    while (num != 0)
    {
        sum += num % 10;
        num /= 10;
    }
    printf("Suma cifara prirodnog broja je: %d\n", sum);
}
```

2. Napisati program koji učitava prirodan broj N i koji ispituje da li je taj broj savršen broj. Prirodan broj je savršen ukoliko je jednak sumi svih svojih djelilaca koji su manji od njega. Na izlazu štampati odgovarajuću poruku.
   Primjer: Broj 28 je djeljiv sa 1,2,4,7,14 i 28, pa je zbir djelilaca manjih od njega 1+2+4+7+14=28, a to znači da je broj 28 savršen.

Program započinjemo deklaracijom promjenljivih i učitavanjem broja N:

```
int N, i, sum = 0;
printf("Unesite broj N: ");
scanf("%d", &N);
```

gdje ćemo i koristiti kao brojač u **for** ciklusu čija je sintaksa:

```
for (init; condition; update)
{
    // naredba ili blok naredbi
}
```

gdje se pomoću izraza **init** postavlja početna vrijednost brojača, dok se pomoću izraza **update** ona ažurira. Potrebno je pronaći sumu djelilaca broja N koji su manji od njega te će početna vrijednost brojača biti 1, a poslednja $N-1$. Na ovaj način ćemo proći kroz sve brojeve od 1 do N i provjeriti da li su oni djelioci broja N i ukoliko jesu dodati ih na sumu. U programskom jeziku C će to biti:

```
for(i = 1; i < N; i++)
{
    if(N%i == 0)
        sum += i;
}
```

Obratiti pažnju da naredba koja prati naredbu uslovnog izvršavanja (**if**) nije ograđena sa zagradama **{}**. Da li je to greška? U kojem slučaju se zagrade mogu izostaviti? Da li su bile neophodne u slučaju ovog for ciklusa? Na kraju programa potrebno je odštampati odgovarajuću poruku, pa je rješenje zadatka:

```c
#include <stdio.h>

int main() {
    int N, i, sum = 0;
    printf("Unesite broj N: ");
    scanf("%d", &N);
    for(i = 1; i < N; i++)
    {
        if(N%i == 0)
            sum +=i ;
    }
    if(sum == N)
        printf("Broj %d jeste savrsen broj\n", N);
    else
        printf("Broj %d nije savrsen broj\n", N);
}
```

Mada je postavkom zadatka naglašeno da je N prirodan broj, šta bi se desilo da je korisnik unio 0? Kako bi se program mogao ,,zaštititi" od ovakvog unosa?

3. Napisati program koji približno računa sumu reda:

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

Sumiranje vršiti sve dok je opšti član sume veći od 10^{-4}.

Pošto je očigledno da će članovi sume biti tipa *double*, promjenljive deklarišemo na sljedeći način:

```c
int n = 1;
double sum = 0, elem;
```

a nakon toga računamo početnu vrijednost člana sume, kada je n jednako 1:

```c
elem = 1.0 / (n * n);
```

Pažnja! ovo je situacija u kojoj se često pravi greška, što nesmotreno, što zbog neznanja. Mada smo deklarisali da je *elem* tipa *double*, nikako ne smijemo napisati sljedeće:

```c
elem = 1 / (n * n);
```

u opštem slučaju. Naime, kako je n tipa *int*, rezultat operacije $n*n$ bio bi takođe *int*. Međutim, dijeljenjem broja 1, koji je u takvom zapisu tipa *int*, i neke vrijednosti koja je takođe tipa *int*, kao rezultat bismo dobili opet *int*. bez obzira što je naša promjenljiva *double* rezultat operacije bio bi *int* zato što se prvo računa međurezultat operacije pa se tako dobijena vrijednost prosljeđuje ka *elem*. U međuvremenu se sve iza tačke odbacuje. Zbog toga je neophodno 1 zapisati kao 1.0 čime se postiže da međurezultat bude takođe tipa *double*. Za povećanje vrijednosti sume korištena je sljedeća sintaksa:

```c
a += b;
```

što rezultuje na isti način kao i naredba:

```c
a = a + b;
```

i predstavlja skraćeni zapis. Konačno, rješenje zadatka je:

```c
#include <stdio.h>

int main()
{
    int n = 1;
    double sum = 0, elem;
    elem = 1.0 / (n * n);
    while(elem > 1e-4){
          sum += elem;
          n++;
          elem = 1.0 / (n * n);
    }
    printf("Suma je %f", sum);
}
```

4. Napisati program koji računa najveći zajednički djelilac (NZD) brojeva a i b pomoću Euklidovog algortima:
   1. Ako je a=b, tada je a=b=NZD i to je kraj algoritma.
   2. Od većeg broja oduzmemo manji i vraćamo se na prvi korak.

Rješenje zadatka je:

```c
#include <stdio.h>

int main()
{
    int a, b;
    puts("Unesite dva cijela broja:");
    scanf("%d %d", &a, &b);
    while(a != b) {
        if (a > b)
            a -= b;
        else
            b -= a;
    }
    printf("NZD je: %d\n", a);
}
```

gdje se za razliku od ostalih zadataka umjesto *printf()* koristila funkcija *puts()* u liniji 7. Razlika je u tome što *puts()* kao argument prima samo tekst na kraju kojeg sama prelazi u novi red, a ne prima druge argumente koji bi eventualno zamijenili čuvare mjesta što je pogodnost funkcije *printf()*.