Workshop – 1. dio

Rekurzivne funkcije

Pogledajte definiciju matematičke operacije faktorijel:

$0! = 1$

$n! = n \cdot (n-1)!$

Primjer: 4! = 4 * 3 * 2 * 1 ili 4! = 4 * 3!

Definicija je veoma prosta i iz nje zaključujemo da je vrijednost faktorijela broja `0` jednaka `1`, kao i da je vrijednost faktorijela od nekog nepoznatog broja `n` jednaka:

`n` * vrijednosti faktorijela broja `(n-1)`

Pogledajte narednu funkciju `int f1(int n)` koja računa faktorijel broja `n`. Oslanjajući se na gornju matematičku definiciju operacije faktorijel, provjerit ćemo da li je `n == 0`.
ako jeste: vratit ćemo sa naredbom `return` vrijednost:
1
ako nije: vratit ćemo vrijednost:
`n` * rezultat funkije `f2` za parametar `(n-1)`.

```
1:   int f1(int n)
2:   {
3:         if (n==0)
4:               return 1;
5:         else
6:               return (n * f2(n-1) );
7:   }
```

Pretpostavimo da funkcija `f2` , također, računa faktorijel parametra funkcije.

Znači, kada iz `main` funkcije pozovemo funkciju `f1` da izračuna faktorijel broja 5, funkcija `f1` će broj 5 primiti kao `n`, zatim će se (pošto je `n>0`) u liniji 6 računati:

```
5  *  faktorijel od 4        // tj.
n  *  faktorijel od (n-1)    // napomena: (n-1)! će izračunati funkcija  f2
```

- uloga funkcije `f1` (u slučaju `n>0`) jeste da pomoću naredbe `return` vrati vrijednost: `n * f2(n-1)`, i ništa više
- funkciju `f1` ,,ne zanima'' kako će funkcija `f2` izračunati vrijednost `n-1`

Ostali smo još ,,dužni'' da definiramo funkciju `f2`:

Znamo da je uloga funkcije `f2` da računa faktorijel nekog broja, a to znači da i za ovu funkciju možemo primijeniti matematičku definiciju faktorijela koju smo prethodno naveli.

- neka ulazna vrijednost funkcije `f2` (tj. formalni parametar) bude varijabla `x`
- pretpostavimo da `f2` treba da izračuna vrijednost faktorijela od `4`, znači `x = 4`

Pošto je x>0, funkcija treba računati:

```
4  *  faktorijel od 3          // tj.
x  *  faktorijel od (x-1)
```

Kako će funkcija f2 izračunati faktorijel od vrijednosti x (tj. 4)?

- Veoma jednostavno: Funkcija f2 će, ako je x>0, vratiti vrijednost: x * faktorijel (x-1). Za računanje faktorijela broja (x-1) koristit ćemo funkciju f1 koju smo već definirali:

```
1:   int f2(int n)
2:   {
3:         if (n==0)
4:               return 1;
5:         else
6:               return (n * f1(n-1) );
7:   }
```

Da bi kompletirali program, moramo još napraviti prototipove funkcija, jer ovdje funkcija poziva drugu funkciju.

Primjer računanja faktorijela broja 3 je prikazan u slijedećem primjeru:

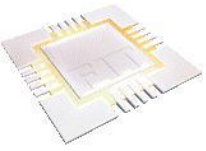
```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int);   // prototip
5:   int f2(int);   // prototip
6:
7:   void main()
8:   {
9:         int r;
10:        r = f1(3);
11:
12:        cout << "3! = " << r << endl;
13:  }
14:
15:  int f1(int n)
     // ...
23:  int f2(int x)
     // ...
```

Računanje faktorijela možete zamisliti na sljedeći način:

- funkcija f1 za n=3 računa samo: 3 * f2(2)
- izračunata vrijednost se vraća kroz return u glavni program i ispisuje se na ekran

Ovdje nije kraj računanja - slijedi detaljnija analiza:

Funkcija f1 za parametar 3 ne može vratiti odgovor dok f2 za 2 ne vratiti odgovor. Ovo možete zamisliti kao slijedeći primjer:
Osoba m (main) pita osobu f1 ,,Koliko je fakorijel od 3?'', a f1 odgovori ,,To je 3 * faktorijel od 2. Ali čekaj da pitam osobu f2 koliko je faktorijel od 2. Tek kad mi osoba f2 izračuna 2!, ja ću ti reći koliko je rješenje za 3!'' Osoba f2 odgovara: ,,Faktorijel od 2 je: 2 * faktorijel od 1. Ali, čekaj dok pitam osobu f1 koliko je faktorijel od 1''.



Ali pazite: Od osobe `f1` su tražena dva odgovora. To možete zamisliti da osobe zapisuju na papir sva pitanja koja su na čekanju, a da ih, pri zapisivanju, numerišu. Na taj način ona zna kome će vratiti odgovor.

Priča se nastavlja: Da bi funkcija `f1` izračunala 1!, ona će pitati osobu `f2` koliko je faktorijel od `0`, a osoba `f2` će odmah odgovoriti „`1`" (*to predstavlja bazni slučaj*), time će osoba `f1` (osobi `f2`) odgovoriti koliko je `1!`. Tako će jedna osoba odgovarati drugoj, ali takvim redoslijedom da se pitanja koja su zadnja postavljena najprije odgovore. Na kraju će osoba `f1` dobiti odgovor na pitanje ,,Koliko je `2!`'', ona će potom taj rezultat pomnožiti sa `3` i tu vrijednost (`6`) reći osobi `m`.

Idemo dalje:
Osobe `f1` i `f2` znaju samo vratiti vrijednost `1` na pitanje ,,0!'' ili pitat drugu osobu, pa tek nakon dobijenog odgovora (od druge osobe) znaju odgovoriti na postavljeno pitanje.

Uočili smo da osobe `f1` i `f2` rade potpuno istu stvar. Stoga je jasno, da osoba `f1` može pitati sama sebe, i nije joj potrebna osoba `f2`. Osoba `f1` će, naravno, davati sama sebi i odgovor. Zamislimo da osoba `f1` svako pitanje, koje sama sebi postavi, zapisat na listić i redati listiće jedan na drugi. Na kraju, kad ne bude bilo potrebe da sama sebe nešto pita, krenut će da ogovara na pitanja koja su zapisana na listiće, uzimajući samo sa vrha, jedan po jedan. Takav način redanja i uzimanja listića sa vrha, ustvari, predstavlja strukturu koja se naziva **stek**.

Zadatak 11:
Prepravite funkciju `f1` tako da, funkcija `f1` ne poziva funkciju `f2` nego funkciju `f1` (samu sebe). Po mogućnosti formatirajte program na sljedeći način:

```
f1(3) = 3 * f1(2)
f1(2) = 2 * f1(1)
f1(1) = 1 * f1(0)
f1(0) = 1
3! = 6
```

Rješenje se nalazi na stranici 13.

Zadatak 12:
Napravite rekurzivnu funkciju za računanje sume kvadrata od `1` do `n`.

Rješenje se nalazi na stranici 13.

Razmislite šta bi osoba `f1` odgovorila na pitanje:

,,Kolika je suma kvadrata od `7`?''
- [redacted]

,,Kolika je suma kvadrata od `1`?''
- [redacted]

Zadatak 13:
Napravite rekurzivnu funkciju za računanje sume kvadrata od `m` do `n`.

a) koristite uzlaznu rekurziju
b) koristite silaznu rekurziju

Rješenje se nalazi na stranici 14.

Razmislite šta bi osoba `f1` odgovorila na pitanje:

Prvi način – uzlazna rekurzija:
,,Kolika je suma kvadrata od `3` do `7`?''

- [redacted] // uzlazna rekurzija

,,Kolika je suma kvadrata od 12 do 12?''
- [redacted] // za `m = n` (bazni slučaj)

Drugi način – silazna rekurzija:
,,Kolika je suma kvadrata od 3 do 7?''
- [redacted] // silazna rekurzija

,,Kolika je suma kvadrata od 12 do 12?''
- [redacted] // za `m = n` (bazni slučaj)

Zadatak 14:

Napravite rekurzivnu funkciju za računanje `n`-tog *Fibonacci* broja.

n:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**...**
n-ti broj:	1	1	2	3	5	8	13	21	34	...

Rješenje se nalazi na stranici 14.

Da vidimo šta bi osoba `f1` odgovorila na pitanje:

,,Koji je prvi ili drugi fibinacci broj (tj. `n≤2`) ?''
- [redacted] // (bazni slučaj)

,,Koji je `7`-ti fibinacci broj (tj. `n=7`) ?''
- [redacted]

Zadatak 15:

Napišite rekurzivnu funkciju koja izračunava sumu neparnih brojeva do zadanog (u glavnoj funkciji) prirodnog broja `n`. U glavnoj funkciji ispišite rezultat. Program riješite na dva načina:

a) neka funkcija `main` provjerava da li je `n` neparan broj

b) neka rekurzivna funkcija provjerava da li je `n` neparan broj

Rješenje se nalazi na stranici 15.

Pomoć za zadatak a:

Funkcija `main` će prosljeđivati, kao aktuelni parametar, funkciji `f1` broj koji mora biti neparan. U tom slučaju bi osoba `f1` odgovorila (pošto ona očekuje da je broj neparan):

,,Kolika je vrijednost sume neparnih brojeva do `0` (`n=0`) ?''
- [redacted] // (bazni slučaj)

,,Kolika je vrijednost sume neparnih brojeva do `7` , tj. do `n` ?''
- [redacted]

Pomoć za zadatak b:

Funkcija `main` će, kao aktuelni parametar, prosljeđivati funkciji `f1` broj koji može biti paran ili neparan.

U tom slučaju bi osoba `f1` odgovorila (pošto ona očekuje da broj može biti i neparan):

,,Kolika je vrijednost sume neparnih brojeva do `0` (`n=0`) ?''
- [redacted] // (bazni slučaj)

,,Kolika je vrijednost sume neparnih brojeva do `8`, tj. ako je `n` paran broj (`n%2==0`) ?''
- ,,To je isto kao suma neparnih brojeva do broja 7 (n-1).''

,,Kolika je vrijednost sume neparnih brojeva do 7, tj. ako je `n` neparan broj ?''
- ,,To je 7 (n) + suma neparnih brojeva do broja 5 (n-2).''

Zadatak 16:

Napišite rekurzivnu funkciju koja izračunava sumu prirodnih brojeva djeljivih sa `3` do zadanog (u glavnoj funkciji) prirodnog broja `n`. U glavnoj funkciji ispišite rezultat.
Program riješite na dva načina:

a) neka funkcija `main` provjerava da li je broj `n` djeljiv sa `3`

b) neka rekurzivna funkcija provjerava da li je broj `n` djeljiv sa `3`

Rješenje se nalazi na stranici 16.

Pomoć - a:

Funkcija `main` će prosljeđivati, kao parametar, funkciji `f1` broj koji mora biti djeljiv sa `3`. U tom slučaju bi osoba `f1` odgovorila (pošto ona očekuje da je broj djeljiv sa `3`):

,,Kolika je vrijednost sume brojeva (djeljivih sa `3`) do broja `1` ili do broja `2` (tj. `n≤2`) ?''
- [redacted] // (bazni slučaj)

,,Kolika je vrijednost sume brojeva (djeljivih sa `3`) do broja `27` (tj. `n=27`) ?''
- [redacted]

Pomoć - b:

Funkcija `main` će prosljeđivati, kao parametar, funkciji `f1` broj koji ne mora biti djeljiv sa `3`. U tom slučaju rekurzivna funkcija treba voditi računa o tome da li je ulazi parametar djeljiv sa brojem `3`.

Osoba `f1` trebala bi odgovoriti (pošto ona očekuje bilo koji prirodan broj):

,,Kolika je vrijednost sume brojeva (djeljivih sa `3`) do broja `1` ili do broja `2` (tj. `n≤2`) ?''
- [redacted] // (bazni slučaj)

,,Kolika je vrijednost sume brojeva (djeljivih sa `3`) do `n`, ako broj `n` nije djeljiv sa 3 (npr. `n=29`) ?''
- ,,To je isto kao suma brojeva djeljivih sa 3 do n-1, tj. 28.''

,,Kolika je vrijednost sume brojeva (djeljivih sa `3`) do `n`, ako je broj `n` djeljiv sa 3 (npr. `n=27`) ?''
- [redacted]

Zadatak 17:

Napravite rekurzivnu funkciju za računanje sume kvadrata od `m` do `n` koristeći metodu dekompozicije.

Rješenje se nalazi na stranici 17.

Pomoć:

Bazni slučaj: Vrijednost sume kvadrata `8` do `8` je: *vrijednost* `8*8`

Rekurzivni poziv: U ovom slučaju ćemo sumu od `m` do `n` rastaviti na dva dijela i sabrati njihove vrijednosti sume kvadrata:

1. sumu kvadrata od `m` do `sr`
2. suma kvadrata od `sr+1` do `n`

Vrijednost `sr` predstavlja cjelobrojnu vrijednost izraza $\frac{m+n}{2}$.

Primjer: Vrijednost `sr` za sumu kvadrata od `2` do `15` je `int(8.5) = 8`

Vrijednost sume kvadrata od `2` do `15` je:
suma kvadrata od `2` do `8` + suma kvadrata od `9` do `15` *(Ovo predstavlja rekurzivni poziv)*

Zadatak 18:

Napravite rekurzivnu funkciju za računanje sume kvadrata od `m` do `n`. Program formatirajte na sljedeći način:

```
Unesite pocetak i kraj niza (pocetak niza mora biti manji od kraja):
2
10
SumaKubova(2, 10) = 8    + SumaKubova(3, 10)
SumaKubova(3, 10) = 27   + SumaKubova(4, 10)
SumaKubova(4, 10) = 64   + SumaKubova(5, 10)
SumaKubova(5, 10) = 125  + SumaKubova(6, 10)
SumaKubova(6, 10) = 216  + SumaKubova(7, 10)
SumaKubova(7, 10) = 343  + SumaKubova(8, 10)
SumaKubova(8, 10) = 512  + SumaKubova(9, 10)
SumaKubova(9, 10) = 729  + SumaKubova(10, 10)
SumaKubova(10, 10) = 1000
============================================
```

Rješenje se nalazi na stranici 17.

Zadatak 19:

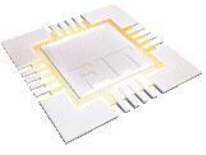
a) Napravite rekurzivnu funkciju koja će računati sumu brojeva koje je korisnik unio. Rekurzivna funkcija treba od korisnika zahtijevati unos brojeva sve dok korisnik ne unese broj `0`.

```
1: 2
2: 5
3: 6
4: 4
5: 0
Suma = 17
```

Rješenje se nalazi na stranici 18.

Pomoć:

- Da li će ova funkcija imati parametre (argumente)?
- Šta su predstavljali parametri za rekurzivnu funkciju u prethodnim zadacima?
- U prethodnim zadacima parametri su određivali broj poziva rekurzivne funkcije, tj. broj poziva funkcije je bio unaprijed poznat.
- Da li je ovdje broj poziva rekurzivne funkcije unaprijed poznat?



b) Dodajte brojač poziva rekurzivne funkcije tako da ispis bude formatiran na dati način. Brojač implementirajte na tri načina:

1) pomoću poziva funkcije po referenci (*pass-by-reference*)
2) pomoću globalne varijable
3) pomoću poziva funkcije po vrijednost (*pass-by-value*)

Rješenje se nalazi na stranici 18.

Zadatak 20:

U funkciji `main` pitajte korisnika koliko brojeva želi unijeti. Zatim definišite rekurzivnu funkciju koja će računati sumu brojeva koje je korisnik unio. Rekurzivna funkcija treba od korisnika zahtijevati unos brojeva sve dok korisnik ne unese broj `0` ili dok ne dostigne broj unosa koji je korisnik odredio u funkciji `main`.

Program riješite na dva načina:

a) pomoću uzlazne rekurzije (brojač rekurzije se uvećava od ... do ...) sa pozivom funkcije po vrijednosti
b) pomoću silazne rekurzije (brojač rekurzije se umanjuje od ... do 0) sa pozivom funkcije po vrijednosti

Rješenje se nalazi na stranici 20.

Pomoć:

Koliko će parametara imati uzlazna rekurzivna funkcija,
a koliko silazna rekurzivna funkcija?

Ovdje ćete naći odgovor.

Kako se određuje broj parametara za rekurzivnu funkciju kod poziva funkcije po vrijednosti?

Zadatak 21:

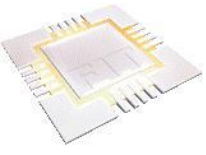
Napravite rekurzivnu funkciju koja će zahtjevati unos pozitivnih brojeva od korisnika sve dok korisnik ne unese negativan broj ili broj 0. Povratna vrijednost iz funkcije treba da najveći uneseni broj.

Rješenje će biti u sljedećem Workshop-u.

Pomoć:

Parametar funkcije je trenutni najveći broj.

Koja će biti vrijednost aktuelnog parametra pri pozivu rekurzivne funkcije iz glavnog programa? - Nula



Rješenja

Rješenje zadatka br. **11**:

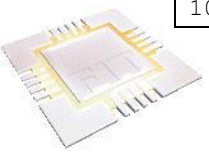
```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int n)
5:   {
6:          cout << "f1(" << n << ") = ";
7:          if (n==0)
8:          {
9:                 cout << 1 << endl;
10:                return 1; // bazni slucaj
11:         }
12:         else
13:         {
14:                cout << n << " * f1(" << n-1 << ") \n";
15:                return (n * f1(n-1) );
16:         }
17:  }
18:
19:  void main()
20:  {
21:         int r;
22:
23:         r = f1(3);
24:         cout << "3! = " << r << endl;
25:  }
```

Ovaj program se izvršava na slijedeći način:

- dok funkcija `main` izvršava liniju broj 24, ona (`main`) poziva funkciju `f1(3)`
- dok funkcija `f1(3)` izvršava liniju broj 15, ona (`f1(3)`) poziva funkciju `f1(2)`
- dok funkcija `f1(2)` izvršava liniju broj 15, ona (`f1(2)`) poziva funkciju `f1(1)`
- dok funkcija `f1(1)` izvršava liniju broj 15, ona (`f1(1)`) poziva funkciju `f1(0)`
- dok funkcija `f1(0)` izvršava liniju broj 10, ona (`f1(0)`) vraća 1 funkciji `f1(1)`
- funkcija `f1(1)` vraća funkciji `f1(2)` vrijednost 1 * 1 = 1
- funkcija `f1(2)` vraća funkciji `f1(3)` vrijednost 2 * 1 = 2
- funkcija `f1(3)` vraća funkciji `main` vrijednost 3 * 2 = 6
- funkcija `main` ispisuje vrijednost 6

Rješenje zadatka br. **12**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int n)
5:   {
6:          if (n==1)
7:                 return 1;
8:          else
9:                 return ( n*n + f1(n-1) );
10:  }
```



```
11:  void main()
12:  {
13:         int m;
14:         cout << "Unesite broj: ";
15:         cin >> m;
16:
17:         cout << f1(m) << endl;
18:  }
```

Rješenje zadatka br. **13a** *(uzlazna rekurzija)*:

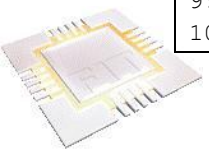
```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int m, int n)
5:   {
6:          if (n==m)
7:                 return m*n;
8:          else
9:                 return ( m*m + f1(m+1, n) );     // uzlazna rekurzija
10:  }
11:
12:  void main()
13:  {
14:         int a, b;
15:         cout << "Unesite broj: ";
16:         cin >> a >> b;
17:
18:         if (a<=b)
19:                cout << f1(a, b) << endl;
20:         else
21:                cout << "Prvi broj mora biti manji od drugog broja \n";
22:  }
```

Rješenje zadatka br. **13b** *(silazna rekurzija)*:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int m, int n)
5:   {
6:          if (n==m)
7:                 return m*n;
8:          else
9:                 return ( f1(m, n-1) + n*n );     // silazna rekurzija
10:  }
```

Rješenje zadatka br. **14**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int fib(int n)
5:   {
6:          if (n<=2)
7:                 return 1;
8:          else
9:                 return ( fib(n-1) + fib(n-2) );
10:  }
```



```
11:
12:  void main()
13:  {
14:         cout << "fib(9) = " << fib(9) << endl;
15:  }
```

Rješenje zadatka br. **15a** *(funkcija `main` provjerava da li je `n` parametar paran broj)*:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int n)
5:   {
6:          if (n <= 0)
7:                 return 0;
8:          else
9:                 return n + f1(n-2);
10:  }
11:
12:  void main()
13:  {
14:         int n;
15:         cout << "Unesite broj: " << endl;
16:         cin >> n;
17:
18:         if (n%2 == 0)   // ako je n paran broj
19:                n--;     // bit će neparan
20:
21:         cout << "f1(" << n << ") = " << f1(n) << endl; //parametar n mora biti neparan broj
22:  }
```

Rješenje zadatka br. **15b** *(rekurzivna funkcija provjerava da li je `n` parametar paran broj)*:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int n)
5:   {
6:          if (n <= 0)
7:                 return 0;
8:          else
9:          {
10:                if (n%2 == 0)
11:                       return f1(n-1);  // ako je n paran broj
12:                else
13:                       return n + f1(n-2); // ako je n neparan broj
14:         }
15:  }
16:
17:  void main()
18:  {
19:         int n;
20:         cout << "Unesite broj: " << endl;
21:         cin >> n;
22:
23:         cout << "f1(" << n << ") = " << f1(n) << endl;
24:  }
```

Rješenje zadatka br. **16a** *(funkcija `main` provjerava da li je broj `n` djeljiv sa 3)*:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int n)
5:   {
6:         if (n <= 2)
7:               return 0;  //bazni slucaj
8:         else
9:               return n + f1(n-3);
10:  }
11:
12:  void main()
13:  {
14:        int n;
15:        cout << "Unesite broj: ";
16:        cin >> n;
17:
18:        while (n%3 != 0) // ako broj n nije djeljiv sa 3
19:              n--;          // onda ga umanjuj dok ne bude djeljiv sa 3
20:
21:        cout << f1(n) << endl; //poziv funkcije za broj n koji mora biti djeljiv sa 3
22:  }
```

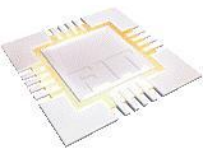
Rješenje zadatka br. **16b** *(rekurzivna funkcija provjerava da li je broj `n` djeljiv sa 3)*:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int n)
5:   {
6:         if (n <= 2)
7:               return 0;  //bazni slucaj
8:         else
9:         {
10:              if (n%3 == 0)
11:                    return n + f1(n-1);
12:              else
13:                    return f1(n-1);
14:        }
15:  }
16:
17:  void main()
18:  {
19:        int n;
20:        cout << "Unesite broj: ";
21:        cin >> n;
22:        cout << f1(n) << endl;
23:  }
```

Rješenje zadatka br. **17**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f1(int m, int n)
5:   {
6:          if (n==m)
7:                 return m*n;
8:          else
9:          {
10:                int sr = (m+n)/2;
11:                return ( f1(m, sr) + f1(sr+1, n) );
12:         }
13:  }
14:
15:  void main()
16:  {
17:         int a, b;
18:         cout << "Unesite broj: \n";
19:         cin >> a >> b;
20:
21:         if (a<=b)
22:                cout << f1(a, b) << endl;
23:         else
24:                cout << "Prvi broj mora biti manji od drugog broja \n";
25:  }
```

Rješenje zadatka br. **18**:



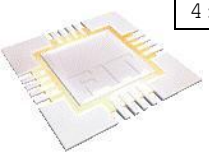
```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int SumaKubova(int m,int n)
5:   {
6:        cout << "SumaKubova(" << m << ", " << n << ") = ";
7:
8:        if (m==n)
9:        {
10:             cout << m*m*m << endl;
11:             return m*m*m;
12:       }
13:       else
14:       {
15:             cout << m*m*m << "\t + SumaKubova(" << m+1 << ", " << n << ") " << endl;
16:             return m*m*m + SumaKubova(m+1,n);
17:       }
18:  }
19:
20:  void main()
21:  {
22:       int m,n;
23:       int r;
24:       cout << "Unesite pocetak i kraj niza (pocetak niza mora biti manji od kraja): \n";
25:       cin >> m >> n;
26:
27:       r = SumaKubova(m,n);
28:       cout << "=============================================\n\n";
29:       cout << "Rezultat sume kubova od " << m << " do " << n << " je: " << r << endl;
30:  }
```

Rješenje zadatka br. **19 - a**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f()
5:   {
6:         int x;
7:         cout << "Unesi broj: ";
8:         cin >> x;
9:
10:        if (x==0)
11:              return 0;
12:        else
13:              return x + f();
14:  }
15:
16:  void main()
17:  {
18:        int s;
19:        s = f();
20:        cout << "Suma = " << s << endl;
21:  }
```

Rješenje zadatka br. **19 – b1**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f(int& b)
```



```
5:   {
6:          b++;
7:          int x;
8:          cout << b << ": ";
9:          cin >> x;
10:
11:         if (x==0)
12:                return 0;
13:         else
14:                return x + f(b);
15:  }
16:
17:  void main()
18:  {
19:         int s;
20:         int b = 0;
21:         s = f(b);
22:         cout << "Suma = " << s << endl;
23:  }
```

Rješenje zadatka br. **19 – b2**:

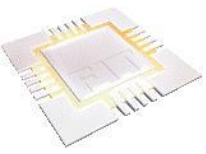
```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int b = 0;
5:
6:   int f()
7:   {
8:          b++;
9:          int x;
10:         cout << b << ": ";
11:         cin >> x;
12:
13:         if (x==0)
14:                return 0;
15:         else
16:                return x + f();
17:  }
18:
19:  void main()
20:  {
21:         int s;
22:
23:         s = f();
24:         cout << "Suma = " << s << endl;
25:  }
```

Rješenje zadatka br. **19 – b3**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f(int b)
5:   {
6:          int x;
7:          cout << b+1 << ": ";
8:          cin >> x;
9:
10:         if (x==0)
11:                return 0;
12:         else
13:                return x + f(b+1);
14:  }
15:
16:  void main()
17:  {
18:         int s;
19:
20:         s = f(0);
21:         cout << "Suma = " << s << endl;
22:  }
```

Rješenje zadatka br. **20 - a**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f(int b, int max)
5:   {
6:          if (b >= max)
7:                 return 0;
8:          else
9:          {
10:                int x;
11:                cout << b+1 <<": ";
12:                cin >> x;
13:
14:                if (x==0)
15:                       return 0;
16:                else
17:                       return x + f(b+1, max);
18:         }
19:  }
20:
21:  void main()
22:  {
23:         int s, max;
24:         cout << "Koliko brojeva zelite unijeti: ";
25:         cin >> max;
26:         cout << endl;
27:
28:         s = f(0, max);
29:         cout << "Suma = " << s << endl;
30:  }
```



Rješenje zadatka br. **20 - b**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int f(int max)
5:   {
6:          if (max <= 0)
7:                 return 0;
8:          else
9:          {
10:                int x;
11:                cout << "Unesi jos " << max << " brojeva: ";
12:                cin >> x;
13:
14:                if (x==0)
15:                       return 0;
16:                else
17:                       return x + f(max-1);
18:         }
19:  }
20:
21:  void main()
22:  {
23:         int s, max;
24:         cout << "Koliko brojeva zelite unijeti: ";
25:         cin >> max;
26:         cout << endl;
27:
28:         s = f(max);
29:         cout << "Suma = " << s << endl;
30:  }
```