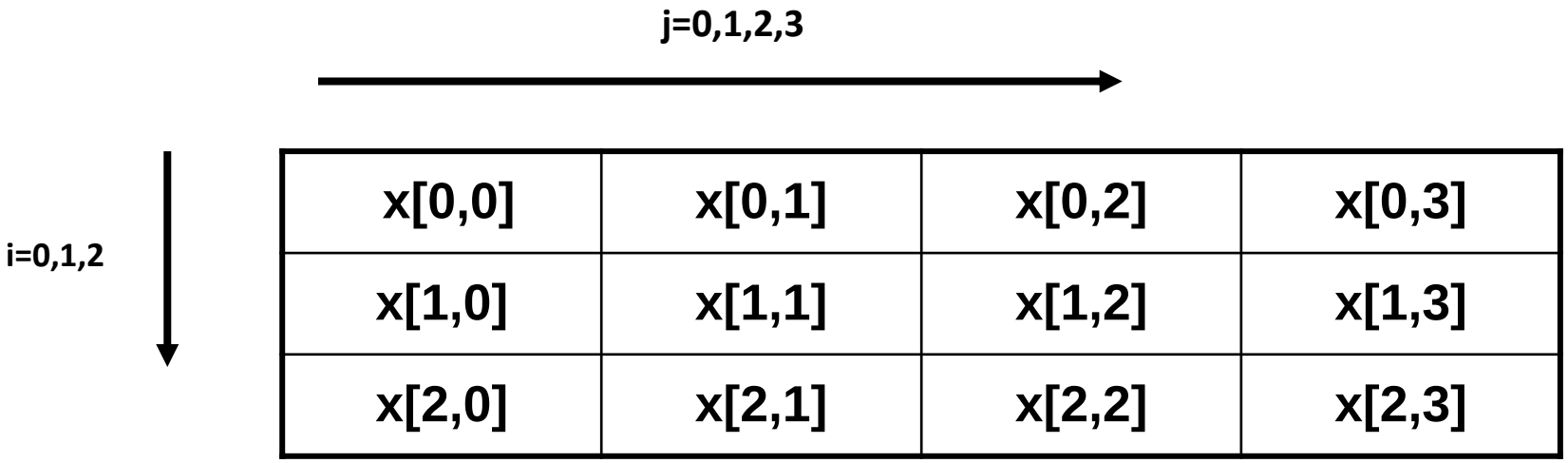
Dvodimenzionalni nizovi

Dvodimenzionalni nizovi (matrice)

```
int[,] x = new int[3, 4];
// deklaracija I instanciranje 2D niza od tri vrste i četiri kolone
```

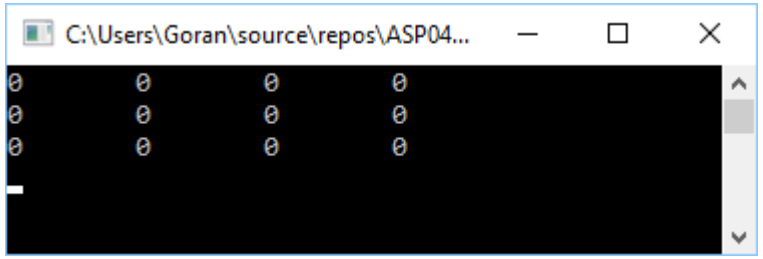


x[0,0]	x[0,1]	x[0,2]	x[0,3]
x[1,0]	x[1,1]	x[1,2]	x[1,3]
x[2,0]	x[2,1]	x[2,2]	x[2,3]

x[i,j] – u preseku i-te vrste i j-te kolone matrice

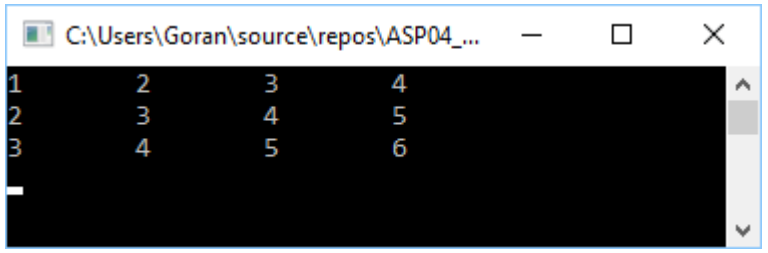
Dvodimenzionalan niz

```
static void Main(string[] args)
{
            int[,] x = new int[3, 4];

            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 4; j++)
                {
                    Console.Write(x[i,j] + "\t");
                }
                Console.WriteLine();
            }
            Console.ReadLine();
}
```



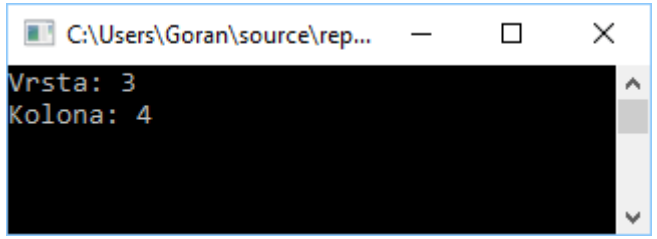
Inicijalizacija 2D niza

```
static void Main(string[] args)
{
    //int[,] x = new int[3, 4];
    int[,] x = {
            {1,2,3,4},
            {2,3,4,5},
            {3,4,5,6}
        };

    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            Console.Write(x[i, j] + "\t");
        }
        Console.WriteLine();
    }
    Console.ReadLine();
}
```



Dimenzije 2D niza

```
static void Main(string[] args)
{
    int[,] x = {
            {1,2,3,4},
            {2,3,4,5},
            {3,4,5,6}
        };

    int brojVrsta = x.GetLength(0);
    int brojKolona = x.GetLength(1);

    Console.WriteLine($"Vrsta: {brojVrsta}");
    Console.WriteLine($"Kolona: {brojKolona}");

    Console.ReadLine();
}
```

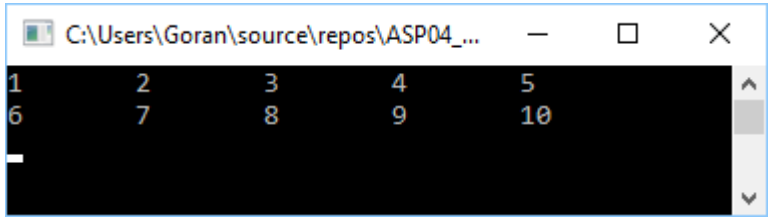


Štampanje članova 2D niza

```
static void Pisi2DNiz(int[,] x)
{
    int brojVrsta = x.GetLength(0);
    int brojKolona = x.GetLength(1);

    for (int i = 0; i < brojVrsta; i++)
    {
        for (int j = 0; j < brojKolona; j++)
        {
            Console.Write(x[i, j] + "\t");

        }
        Console.WriteLine();
    }
}
```

Poziv metode za štampanje

```
static void Main(string[] args)
{
    int[,] x1 =  {
            { 1, 2, 3, 4, 5 },
            { 6, 7, 8, 9, 10 }
        };
    Pisi2DNiz(x1);
    Console.ReadLine();
}
```



Metoda za učitavanje 2D niza

```
static void Citaj2DNz(int[,] x)
{
    int brojVrsta = x.GetLength(0);
    int brojKolona = x.GetLength(1);

    for (int i = 0; i < brojVrsta; i++)
    {
        for (int j = 0; j < brojKolona; j++)
        {
            Console.WriteLine($"x[{i},{j}]=?");
            x[i, j] = int.Parse(Console.ReadLine());
        }
    }
}
```

Iscrtavanje linije na konzoli

```
static void Linija(int n)
{
    //iscrtava liniju  duzine n na konzoli
    Console.WriteLine("".PadRight(n, '_'));
}
```

Učitavanje i štampanje matrice

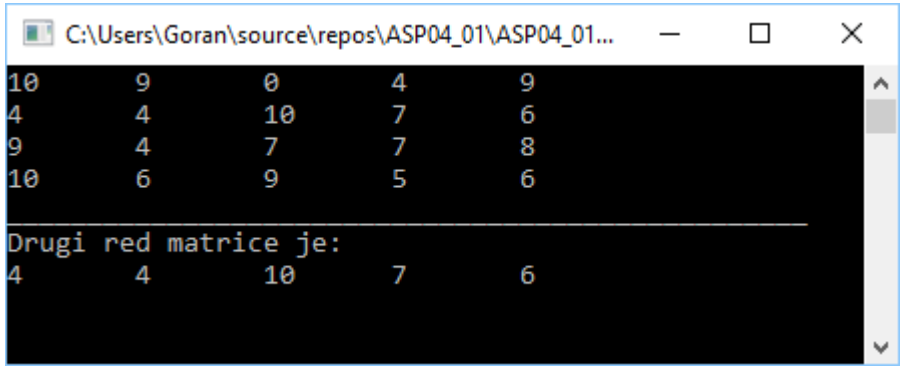
```
static void Main(string[] args)
{
    Console.WriteLine("Unesi broj redova matrice:");
    int n = int.Parse(Console.ReadLine());

    Console.WriteLine("Unesi broj kolona matrice:");
    int m = int.Parse(Console.ReadLine());

    int[,] x = new int[n, m];

    Citaj2DNz(x);

    Linija(50);

    Pisi2DNiz(x);

    Console.ReadLine();
}
```

Metoda za štampanje jednodimenzionalnog niza

```
static void PisiNiz(int[] x)
{
    foreach (int i in x)
    {
        Console.Write(i + "\t");
    }
    Console.WriteLine();
}
```

Kreiranje 1D niza od reda 2D niza

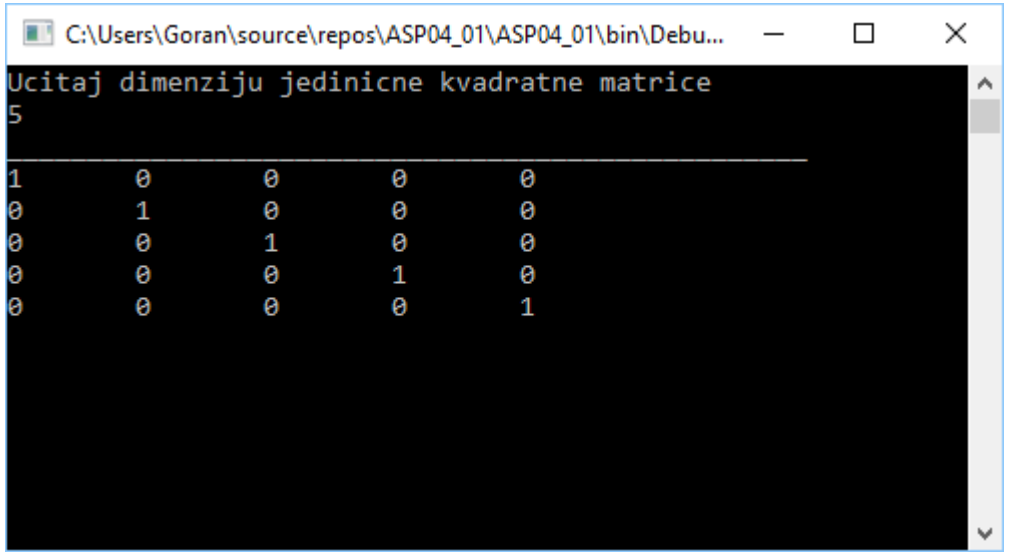
```
static void Main(string[] args)
{
    // matrica od 4 reda i 5 kolone
    int[,] x = new int[4, 5];

    Random rnd = new Random();
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            x[i, j] = rnd.Next(11);
        }
    }
    Pisi2DNiz(x);
    Linija(50);

    Console.WriteLine("Drugi red matrice je:");

    int[] x1 = new int[5];
    for (int i = 0; i < 5; i++)
    {
        // prepisujemo drugi red u niz
        x1[i] = x[1, i];
    }

    PisiNiz(x1);
    Console.ReadLine();
}
```



Primer 1

Napisati program za prikazivanje na konzoli jedinične kvadratne matricije dimenzije nxn

```
static void Main(string[] args)
{
    Console.WriteLine("Ucitaj dimenziju jedinicne kvadratne matrice");
    int n = int.Parse(Console.ReadLine());

    int[,] x = new int[n, n];

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
            {
                x[i, j] = 1;
            }
            else
            {
                x[i, j] = 0;
            }
        }
    }
    Linija(50);
    Pisi2DNiz(x);
    Console.ReadLine();
}
```



Primer 2

Odreditu sumu elemenata na glavnoj dijagonali matrice dimenzija **n x m** .

```
static void Main(string[] args)
    {

        Console.WriteLine("Unesi broj redova matrice:");
        int n = int.Parse(Console.ReadLine());

        Console.WriteLine("Unesi broj kolona matrice:");
        int m = int.Parse(Console.ReadLine());

        int[,] x = new int[n, m];

        Citaj2DNz(x);

        Linija(50);
        Pisi2DNiz(x);
        Linija(50);

        int suma = 0;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
            {
                if (i == j)
                {
                    suma += x[i,j];
                }
            }
        }

        Console.WriteLine($"Suma elemenata na glavnoj dijagonali je: {suma}");
        Console.ReadLine();
    }
}
```

Rekurzivni algoritmi

Iterativni i rekurzivni algoritmi

- Iterativni algoritam: ponavljanje tela petlje više puta
- Iteracija je postupak koji se primenjuje nad skupom naredbi
- **Rekurzivni** algoritam: ponavljanje samog algoritma više puta
- **Rekurzija** je proces koji se primenjuje nad funkcijom
- **Rekurzivna** funkcija je funkcija koja poziva sama sebe
- Iterativni algoritmi su brži od rekurzivnih algoritama
- Rekurzija smanjuje količinu koda i nekada su rekurzivni algoritmi intuitivniji od iterativnih algoritama

Memorijski zahtevi

- Lokalne promenljive svake funkcije(iterativne i rekurzivne) se smeštaju na stack memoriji
- Kada se pozove funkcija lokalne promenljive pozivajuće funkcije se smeštaju na stack
- Kada se poziv završi, pozivajuća funkcija obnavlja vrednosti lokalnih promenljivih sa steka
- Rekurzivnim rešenjima se formira niz primeraka funkcije sa svojim lokalnim promenljivama što zahteva više memorijskog prostora i vremena potrebnog za njihovo izvršavanje

Iterativni algoritam za pronalažen n-tog stepena broja a

$$a^n = a \cdot a \cdot a \ldots . \cdot a$$

n - množenja

```
static decimal Stepen(decimal a, int n)
{
    decimal rezultat = 1;
    for (int i = 0; i < n; i++)
    {
        rezultat *= a;
    }
    return rezultat;
}
```

Rekurzivni algoritam za pronalažen n-tog stepena broja a

a^0=1

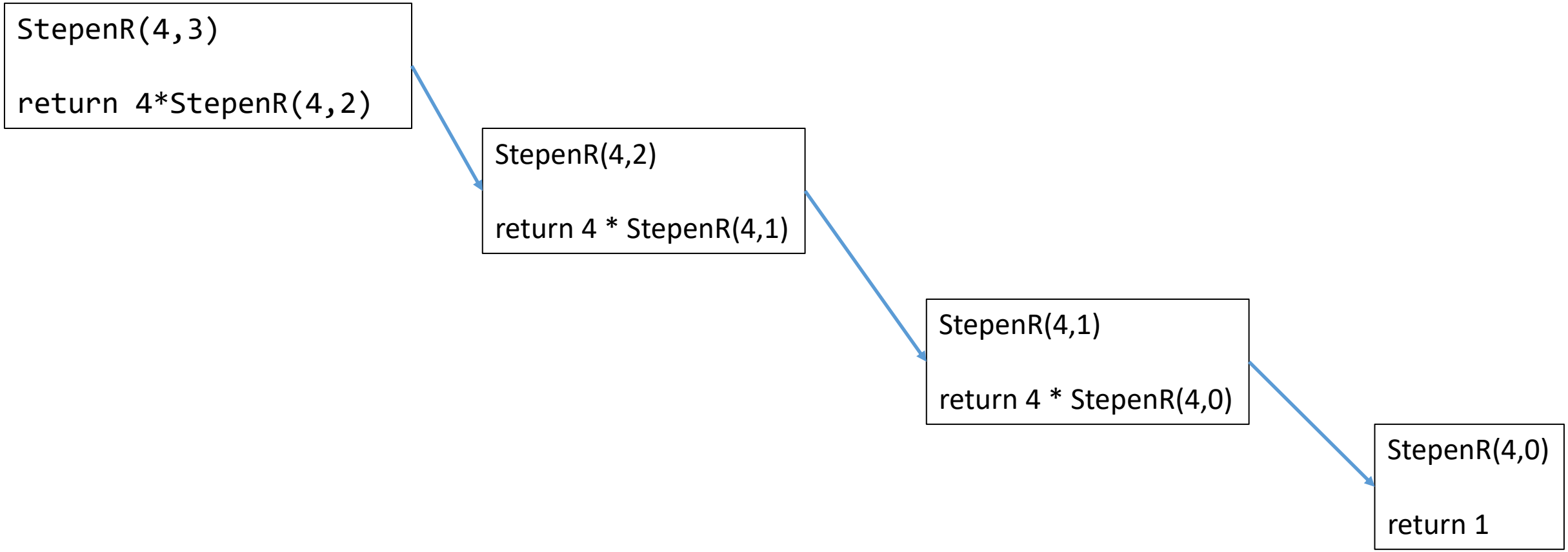
$a^n = a^{n-1} \cdot a$

$f(n) = a^n$

$f(n) = f(n-1) \cdot a$

```
static decimal StepenR(decimal a, int n)
{
    if (n == 0)
    {
        return 1;
    }
    else
    {
        return StepenR(a, n - 1) * a;
    }
}
```

Izvršavanje rekurzivnog algoritma



Rekurzivno računanje sume

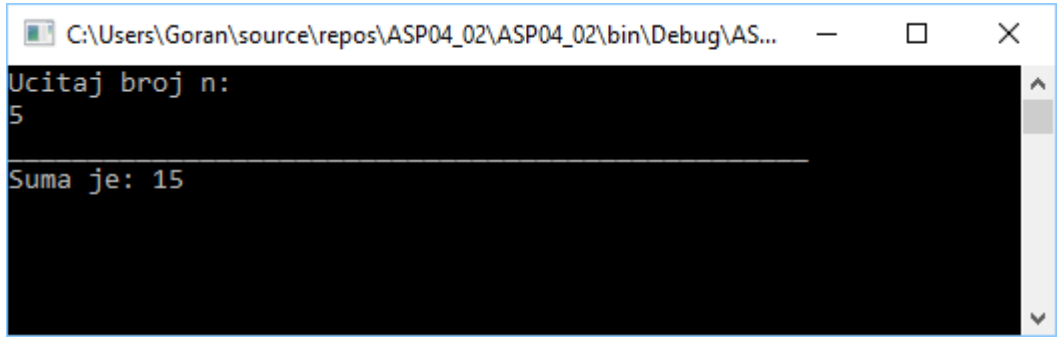
Napisati rekurzivnu funkciju za izračunavanje sume prvih **n** prirodnih brojeva:

S(n-1) = 1 + 2 + ….+ (n-1)
S(n) = 1 + 2 + ….. + (n-1) + n = n + S(n-1)

S(2) = 2 + S(1)
S(1) = 1 + S(0)
S(0) = 0

```
static int Suma (int n)
{
    if (n == 0)
    {
        return 0;
    }
    else
    {
        return n + Suma(n - 1);
    }
}
```

Poziv rekurzivne funkcije



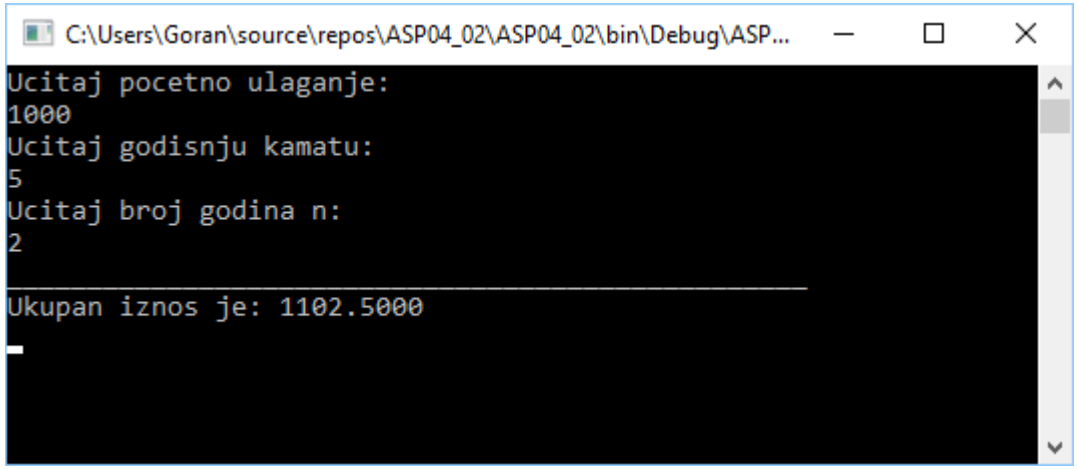
Rekurzivno računanje kamate

Banka na početku svake godine ulagačima obračunava kamatu od **p** procenata na novčanu sumu koja je odležala prethodne godine. Napisati program kojim se izračunava novčani iznos kojim raspolaže ulagač posle **n** godina, ako je početno ulaganje **s**.

a(0) = s	početno ulaganje
a(1) = a(0)+ a(0)* p/100	iznos posle prve godine
a(2) = a(1) + a(1) * p/100	iznos posle druge godine
a(i) = a(i-1) + a(i-1) * p/100	
a(i) = (1 + p/100) * a(i-1)	

```
static decimal Iznos(int n, decimal s, decimal p)
{
    if (n>0)
    {
        return (1+p/100)* Iznos(n - 1, s, p);
    }
    else
    {
        return s;
    }
}
```

Poziv rekurzivne funkcije za računanje kamate



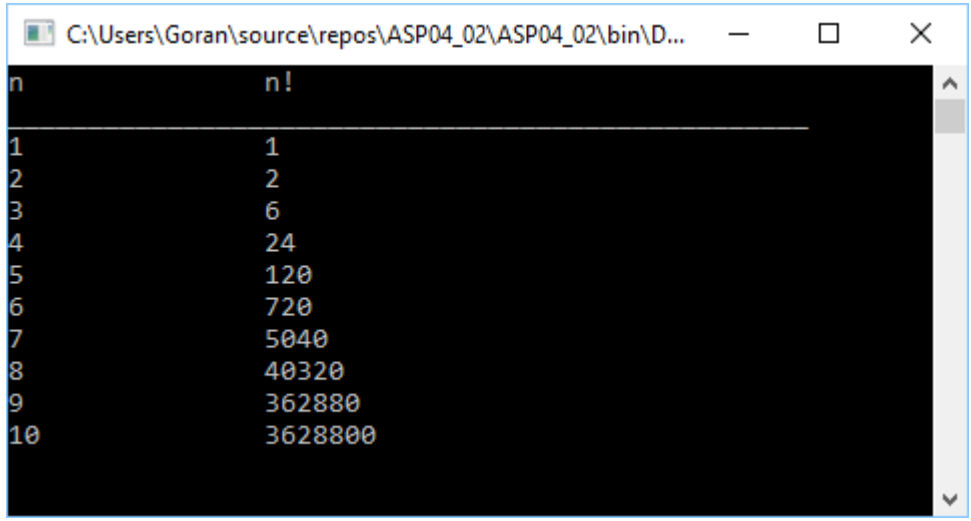
Faktorijel broja

n! = 1 ako je n =1
n! = (n-1)! * n, ako jen >1

```
static long Faktorijel(int n)
{
    if (n > 1)
    {
        return n * Faktorijel(n - 1);
    }
    else
    {
        return n;
    }
}
```

Računanje faktorijela

```
static void Main(string[] args)
{
    Console.WriteLine("n\t\tn!");
    Linija(50);
    for (int i = 1; i <= 10; i++)
    {
        Console.WriteLine($"{i}\t\t{Faktorijel(i)}");
    }

    Console.ReadLine();
}
```



Fibonačijev niz

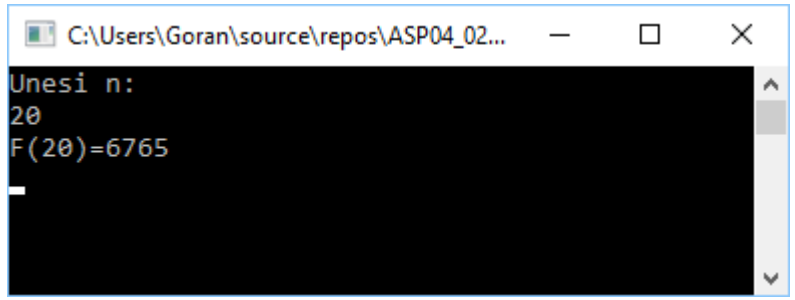
Napisati rekurzivnu funkciju za izračunavanje n-tog člana Fibonačijevog niza:
f(1) =1, f(2) =1
f(3) = f(1) + f(2) = 2
f(4) = f(2) + f(3) = 1 + 2 = 3

f(n) = f(n-2) + f(n-1), za n >2

```
static int Fibonaci(int n)
{
    if (n<3)
    {
        return 1;
    }
    else
    {
        return Fibonaci(n - 2) + Fibonaci(n - 1);
    }
}
```

Računanje n-tog člana Fibonačijevog niza

```
static void Main(string[] args)
{
    Console.WriteLine("Unesi n:");
    int n = int.Parse(Console.ReadLine());
    int clan = Fibonaci(n);

    Console.WriteLine($"F({n})={clan}");

    Console.ReadLine();
}
```



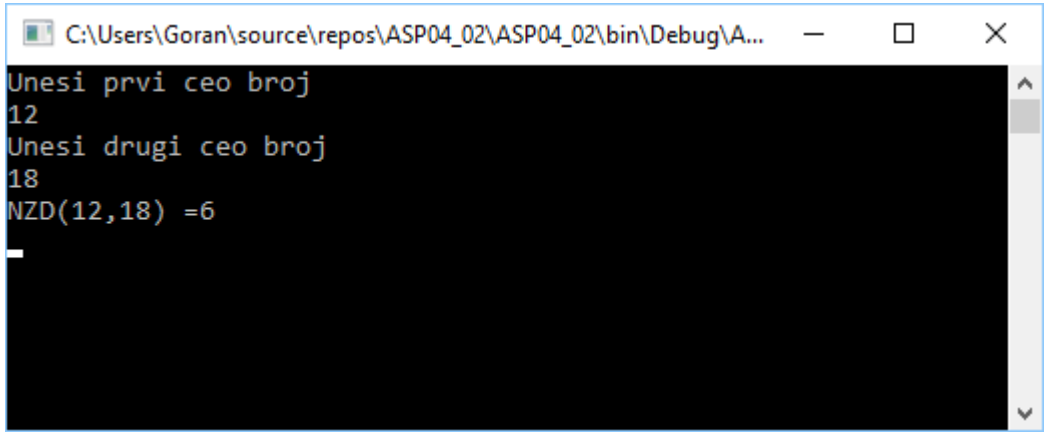
NZD dva broja

Najveći zajednički delilac **NZD** dva cela broja različita od nule je najveći ceo broj koji deli oba broja bez ostatka

```
static int Nzd1(int m, int n)
{
    // manji od brojeva smestimo u d
    int d = m < n ? m : n;

    while (n % d != 0 || m % d != 0)
    {
        d--;
    }
    return d;
}
```

Računanje NZD dva broja

```
static void Main(string[] args)
{
    Console.WriteLine("Unesi prvi ceo broj");
    int n = int.Parse(Console.ReadLine());

    Console.WriteLine("Unesi drugi ceo broj");
    int m = int.Parse(Console.ReadLine());
    int nzd = Nzd1(n, m);
    Console.WriteLine($"NZD({n},{m}) ={nzd}");
    Console.ReadLine();
}
```



Euklidov algoritam za nalaženje NZD dva broja

NZD(m,n) = n, ako je m=0
NZD(m,n) = NZD(n%m,m), ako je m<>0

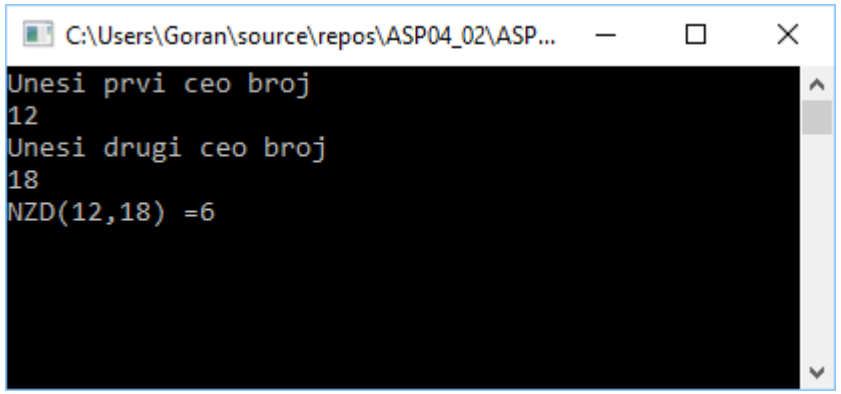
NZD(30,18) = NZD(18%30,30) = NZD(18,30) = NZD(30%18,18) = NZD(12,18) = NZD(18%12,12) = NZD(6,12) = NZD(12 %6, 6) = NZD(0,6) = 6

```
public static int Euklid(int m, int n)
{
    if (m == 0)
    {
        return n;
    }
    else
    {
        return Euklid(n%m,m);
    }
}
```

Ako prvi broj nije manji od drugog, posle prvog poziva funkcije razmenjuju mesta

Poziv Euklidovog algoritma

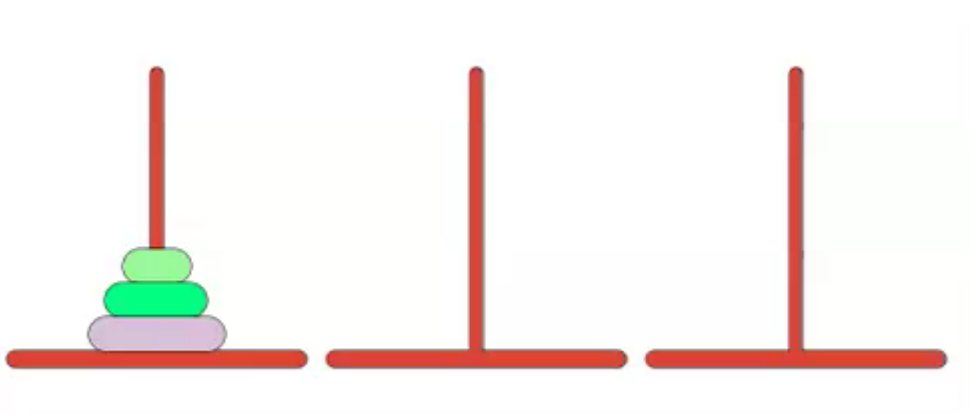
```
static void Main(string[] args)
{
    Console.WriteLine("Unesi prvi ceo broj");
    int m = int.Parse(Console.ReadLine());

    Console.WriteLine("Unesi drugi ceo broj");
    int n = int.Parse(Console.ReadLine());

    int nzd = Euklid(m, n);
    Console.WriteLine($"NZD({m},{n}) ={nzd}");
    Console.ReadLine();
}
```



Igra Hanojska kula

Na stolu se nalaze 3 stuba. Na stubu 1 se nalazi n diskova čiji su poluprečnici opadajući kako se ide ka vrhu. Napisati algoritam kojim se diskovi sa stuba 1 premeštaju na stub 3 uz korišćenje pomoćnog stuba 2, tako da se u jednom koraku premešta jedan disk. Nije dozvoljeno premeštanje većeg diska preko manjeg.

Hanojska kula sa tri diska



Hanojska kula

Ukupan broj premeštanja je: 2^3-1

$n\ diskova$: $broj\ premeštanja\ \ je\ 2^n$-1

Algoritam

- **Prebaci(n, izvor, odrediste, pom)** - prebaci n diskova sa stuba izvor na stub odrediste korišćenjem pomoćnog diska pom
- **Prebaci(n-1, izvor, pom, odrediste)**
- **Prebaci disk sa stuba izvor na stub odrediste**
- **Prebaci(n-1, pom, odrediste, izvor)**

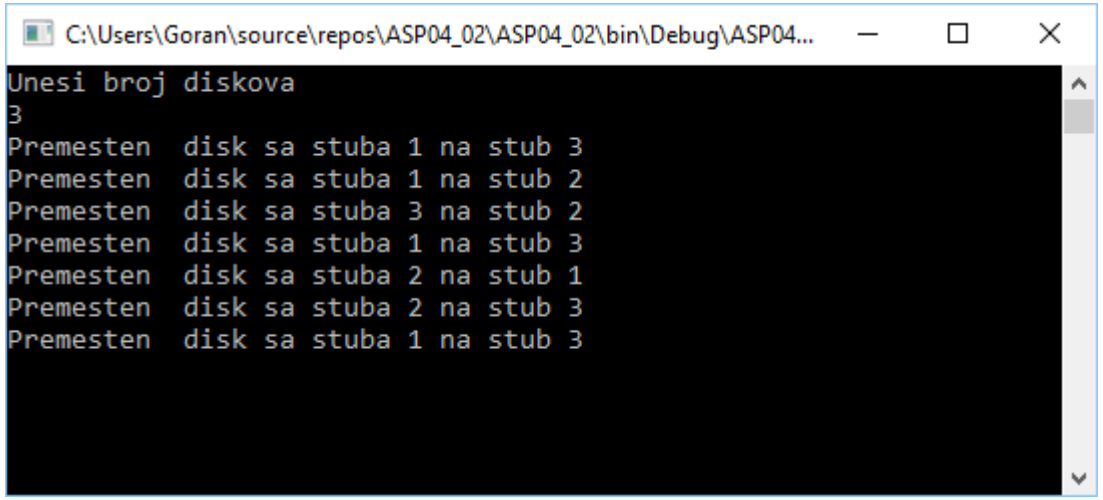
- Prebaci (3,1,3,2) - prebaci 3 diska sa stuba 1 na stub 3 korišćenjem pomoćnog diska 2
- Prebaci (2,1,2,3) - prebaci 2 diska sa stuba 1 na stub 2, pomoćni disk je 3
- Prebaci disk sa stuba 1 na stub 3
- Prebaci (2,2,3,1) - prebaci 2 diska sa stuba 2 na stub 3 korišćenjem pomoćnog diska 1

Realizacija algoritma

```
static void Prebaci(int n, int izvor, int odrediste, int pom)
{
    if (n>0)
    {
        Prebaci(n - 1, izvor, pom, odrediste);
        Console.WriteLine($"Premešten  disk sa stuba {izvor} na stub {odrediste} ");
        Prebaci(n - 1, pom, odrediste, izvor);
    }
}
```

Testiranje algoritma

```
static void Main(string[] args)
{
    Console.WriteLine("Unesi broj diskova");
    int n = int.Parse(Console.ReadLine());
    Prebaci(n, 1, 3, 2);
    Console.ReadLine();
}
```



Pitanje 1

Dvodimenzionalni niz koji podatke čuva u 3 vrste i 5 kolona instancira se na sledeći način:

a. int[,] x = new int[3, 5];
b. int[x] = new int(3,5);
c. int x[] = new int[3,5];

Odgovor: a

Pitanje 2

Šta se dobija izvršavanjem sledećeg koda:

```
static void Main(string[] args)
{
    int[,] x = {
            {1,2,3,4},
            {2,3,4,5},
            {3,4,5,6}
        };

    Console.WriteLine(x[0, 3]);
    Console.ReadLine();
}
```

a. 2
b. 3
c. 4

Odgovor: c

Pitanje 3

Rekurzivna funkcija je funkcija koja
a. Poziva neku drugu funkciju
b. Poziva samu sebe
c. Poziva isključivo iz Main() funkcije

Odgovor: b

Pitanje 4

Data je sledeća rekurzivna funkcija:

```
static int Fun1(int n)
{
   if (n == 4)
   {
      return n;
   }
   else
   {
      return 2 * Fun1(n + 1);
   }
}
```

Koliko je Fun1(2) ?

a. 6
b. 16
c. 24

Odgovor: b

Pitanje 5

Fibonačije niz je:

a. 1,2,3,5,8...
b. 1,1,2,3,5,...
c. 1,3,5,7,9,...

Odgovor: b

Pitanje 6

Prema Euklidovom algoritmu za pronalaženje NZD dva pozitivna cela broja tačno je tvrđenje:

a. NZD(12,18) = NZD(6,12)
b. NZD(12,18) = NZD(30,12)
c. NZD(12,18) = NZD(2,12)

Odgovor: a

Pitanje 7

Minimalni broj premeštanja diskova kod igre Hanojska kula je:

a. 2n-1
b. 2^n-1
c. 2n+1

Odgovor: b