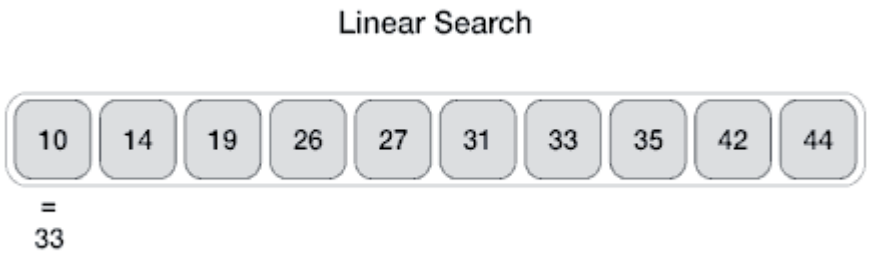
Algoritmi pretrage

Linearna pretraga

- Naziva se još i sekvencijalna pretraga
- Počinje od početka niza
- Sekvencijalno se proveravaju element niza jedan po jedan
- Kraj pretrage
  - Pronađen željeni element - uspešna pretraga, vraćemo indeks prvog pojavljivanja željenog elementa
  - Nije pronađen željeni element - neuspešna pretraga, vraćamo rezultat -1



Implementacija algoritma za linearnu pretragu

```
static int LinSearch(int[] x, int a)
{
    int n = x.Length;

  1  for (int i = 0; i < n; i++)
    {
  2      if (x[i] == a)
        {
  3          return i;
        }
    }
  4  return -1;
}
```

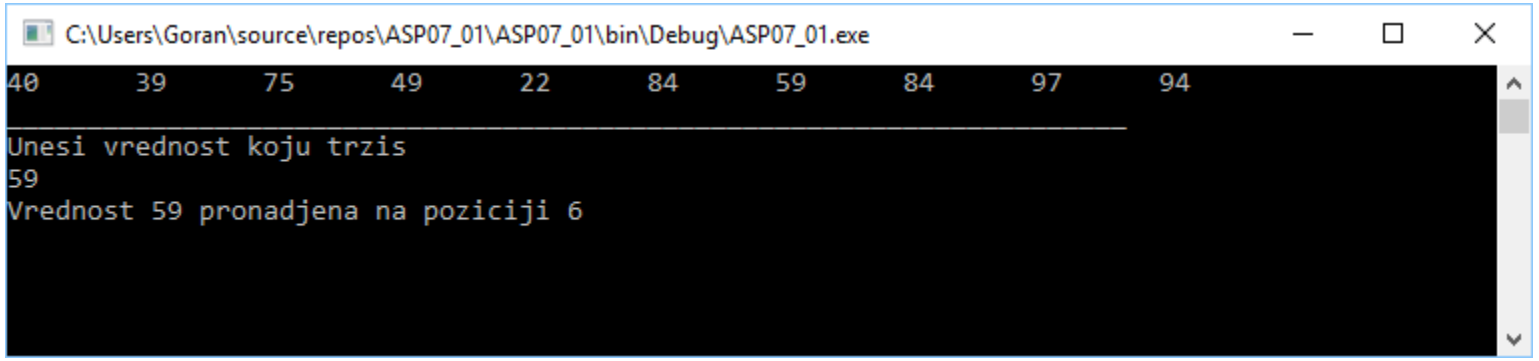
Red	1	2	4
Vreme izvršavanja	n	n	1

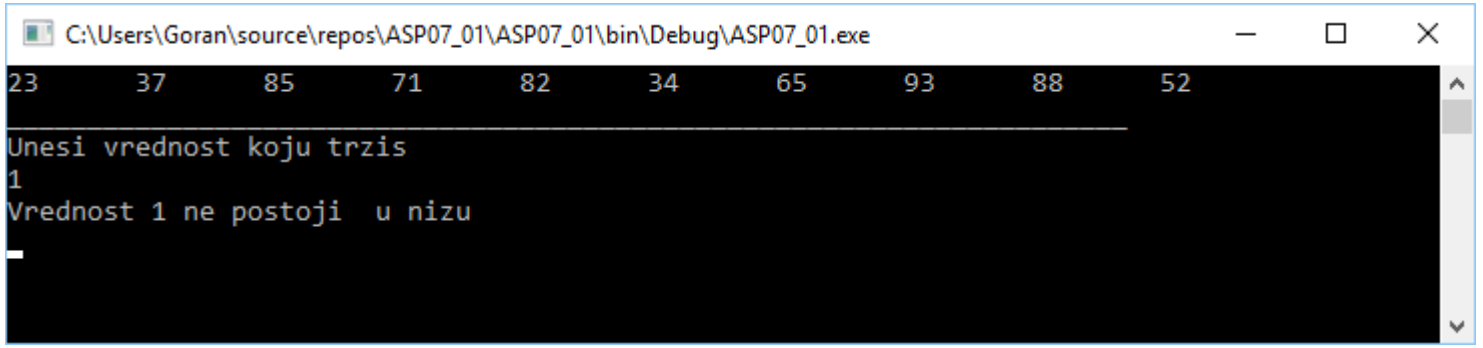
T(n) = 2n+1
O(n) = n

Poziv algoritma za sekvencijalnu pretragu

```
static void Main(string[] args)
{
    int[] x = KreirajNiz(10);
    PisiNiz(x);
    Linija(70);

    Console.WriteLine("Unesi vrednost koju trzis");
    int a = int.Parse(Console.ReadLine());

    int indeks = LinSearch(x, a);

    if (indeks > -1)
    {
        Console.WriteLine($"Vrednost {a} pronadjena na poziciji {indeks}");
    }
    else
    {
        Console.WriteLine($"Vrednost {a} ne postoji  u nizu");
    }

    Console.ReadLine();
}
```

Prikaz rezultata pretrage





Analiza linearne pretrage

- Najbolji slučaj: tražena vrednost se nalazi na početku niza
  - Samo jedno poređenje
  - Vremenska komplesnost $O(1)$
- Najgori slučaj: tražena vrednost nije u nizu
  - Broj poređenja je n
  - Vremenska kompleksnost je $O(n)$
- Prosečan slučaj: tražena vrednost se nalazi na poziciji i
  - Broj poređenja i
  - Prosečan broj poređenja je $P = \frac{1}{n}\ (1\ +\ 2\ + 3 +\ \ldots . + n) = \frac{1}{n}\frac{n}{2}\ (n + 1) = \frac{n+1}{2}$
  - Vremenska kompleksnost je $\mathrm{T(n)} = O(n)$

Linearna pretraga korišćenjem stražara (sentinel)

- Čuvamo poslednji element niza u pomoćnoj promenljivoj
- Vrednost koju tražimo ubacijemo u poslednji element niza
- Petljom while se proverava indeks prvog elementa u listi koji je jednak traženom elementu
- Ponovo se na poslednje mesto niza vraća originalni element
- Proverava se da li je nadjeni indeks manji od (n-1) ili je vrednost elementa na poziciji nađenog indeksa jednaka poslednjem elementu
- Smanjuje se broj poređenja sa kojima radi algoritam na n + 2 u najgorem slučaju

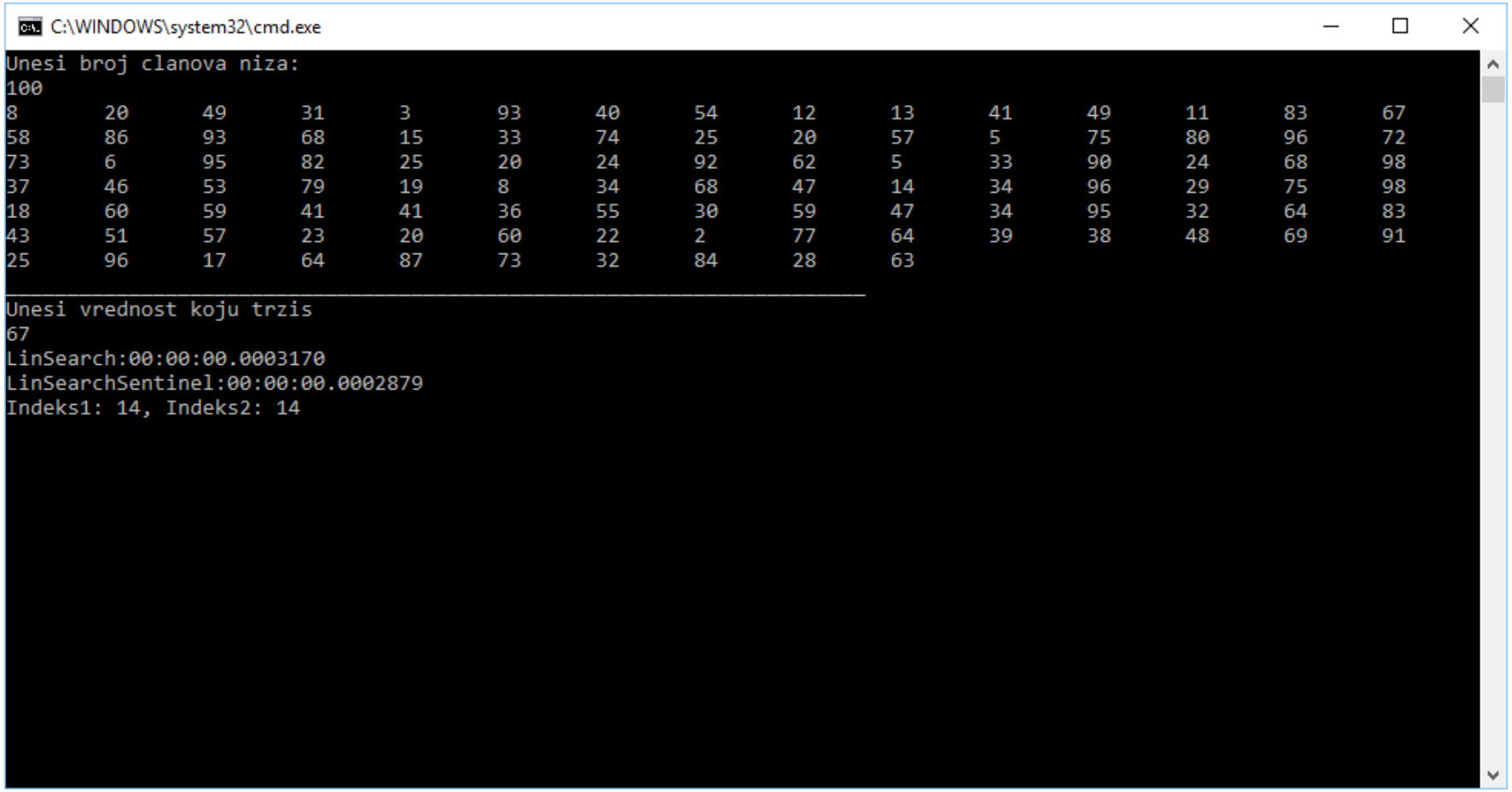
Linearna pretraga korišćenjem stražara - implementacija

```
static int LinSearchSentinel(int[] x, int a)
{
    int n = x.Length;
    int poslednji = x[n - 1];
    x[n - 1] = a;
    int i = 0;
    while (x[i] != a)
    {
        i++;
    }
    x[n - 1] = poslednji;

    if ((i < n - 1) || (a == x[n - 1]))
    {
        return i;
    }
    else
    {
        return -1;
    }
}
```

Esperimentalno upoređivanje algoritama za linearnu pretragu

```
static void Main(string[] args)
{
    Console.WriteLine("Unesi broj clanova niza: ");
    int n = int.Parse(Console.ReadLine());
    int[] x = KreirajNiz(n);
    PisiNiz(x);
    Linija(70);

    Console.WriteLine("Unesi vrednost koju trzis");
    int a = int.Parse(Console.ReadLine());

    Stopwatch t1 = new Stopwatch();
    t1.Start();
    int indeks1 = LinSearch(x, a);
    t1.Stop();
    TimeSpan vreme1 = t1.Elapsed;
    t1.Reset();


    t1.Start();
    int indeks2 = LinSearchSentinel(x, a);
    t1.Stop();

    TimeSpan vreme2 = t1.Elapsed;

    Console.WriteLine($"LinSearch:{vreme1}");
    Console.WriteLine($"LinSearchSentinel:{vreme2}");

    Console.WriteLine($"Indeks1: {indeks1}, Indeks2: {indeks2}");


    Console.ReadLine();
}
```

Rezultat eksperimenta



Algoritam za linearnu pretragu sortiranog niza

```
static int LinSearchSort(int[] x, int a)
{
    int i =0;
    int n = x.Length;
    for ( i = 0; i < n; i++)
    {
        if (x[i] >=a)
        {
            break;
        }
    }

    if (x[i] == a)
    {
        return i;
    }
    else
    {
        return -1;
    }
}
```

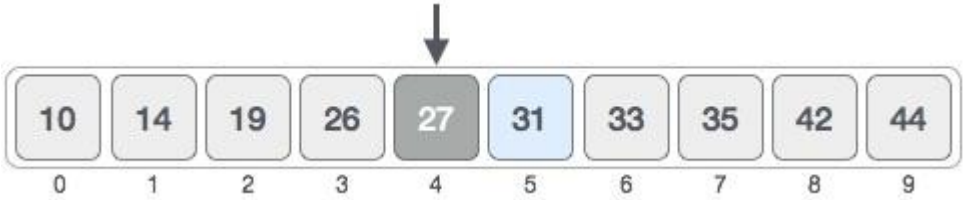
- Poboljšava vreme pretrage kada se vrednost koju tražimo ne nalazi u nizu
- Ne mora se pretražiti ceo niz kao u slučaju nesortiranog niza.

Binarna pretraga -1

traži se broj a=31

Niz sa kojim radimo mora biti sortiran.

donja =0;
gornja =9
sredina = (donja + gornja)/2 = (9+0)/2 = 4 - zaokružujemo na manju vrednost



x[4] =27 < a

ako je a == x[sredina], pronađena vrednost, kraj pretrage
ako je a>x[sredina] , pretražuje se desni podniz
ako je a<x[sredina], pretražuje se levi podniz

Binarna pretraga -2

traži se broj a=31

sredina =4; // stara vrednost
donja =sredina +1 =5; // kada se pretražuje desni podniz
gornja =9
sredina = (donja + gornja)/2 = (5+9)/2 = 7

x[7] =35 > a

Binarna pretraga -3

traži se broj a=31

sredina =7; // stara vrednost
donja = 5;
gornja =sredina -1 =6 // kada se pretražuje levi podniz
sredina = (donja + gornja)/2 = (5+6)/2 = 5

x[5] =31 = a

Implementacija algoritma

```
static int BinSearch(int[] x, int a)
{
    int n = x.Length;
    int gornja = n - 1;
    int donja = 0;
    int sredina;

    while (donja <= gornja)
    {
        sredina = (donja + gornja) / 2;

        if (a == x[sredina])
        {
            return sredina;
        }
        else if (a < x[sredina])
        {
            // pretrazujem levi podniz
            gornja = sredina - 1;
        }
        else
        {
            // pretrazujem desni podniz
            donja = sredina + 1;
        }
    }
    return -1;
}
```

Analiza algoritma binarne pretrage -1

- Na početku 1. iteracije, dužina oblasti pretrage iznosi n
- Na početku 2. iteracije, dužina oblasti pretrage iznosi približno $\frac{n}{2}$
- Na početku 3. iteracije, dužina oblasti pretrage iznosi približno $\frac{\frac{n}{2}}{2} = \frac{n}{4}$
- Na početku poslednje m-te iteracije, dužina oblasti pretrage iznosi približno $\frac{n}{2^{m-1}}$

Analiza algoritma binarne pretrage -2

$$\frac{n}{2^{m-1}}=1$$

$$m-1=\log_2 n$$

$$m=\log_2 n+1$$

$$T(n)\approx m=\log_2 n$$

za n =1000 broj poređenja nije veći od 10

Vremenska kompleksnost algoritma binarne pretrage je $\boldsymbol{T(n) = O(log\, n)}$

Pitanje 1

Vremenska kompleksnost algoritma linearnog pretraživanja je:

a. O(n)
b. O(log n)
c. $O(n^2)$

Odgovor: a

Pitanje 2

Vremenska kompleksnost algoritma binarnog pretraživanja je:

a. O(n)
b. O(log n)
c. $O(n^2)$

Odgovor: b

Pitanje 3

Algoritam binarnog pretraživanja:
a. zahteva da podaci prethodno budu sortirani
b. ne zahteva da podaci prethodno budu sortirani

Odgovor: a