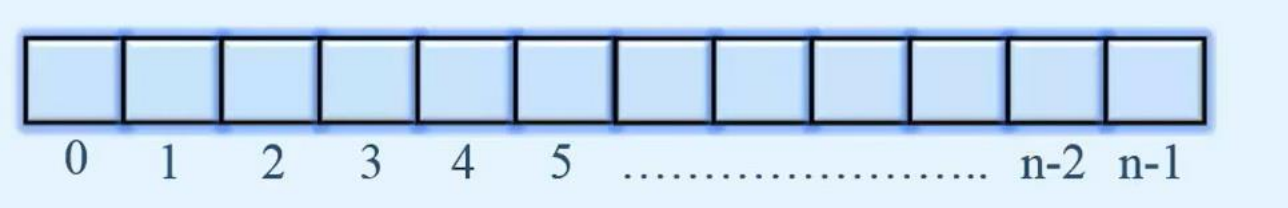
Algoritmi sortiranja-1

Problem sortiranja

- Ako je dat niz neuređenih brojeva, treba preurediti brojeve tog niza tako da oni obrazuju neopadajući niz
- a[0],a[1],....,a[n-1] inicijalni nesortirani niz
- a[0] <=a[1]<==,...<=a[n] sortirani niz
- Sortiranjem se postiže brže pronalaženje informacija nego u slučaju nesortiranog skupa podataka

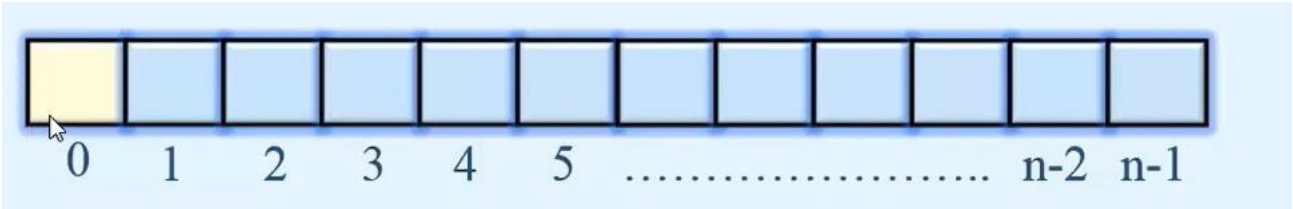
Algoritam Selection Sort

n elemenata: x[0],x[1],....,x[n-2],x[n-1]



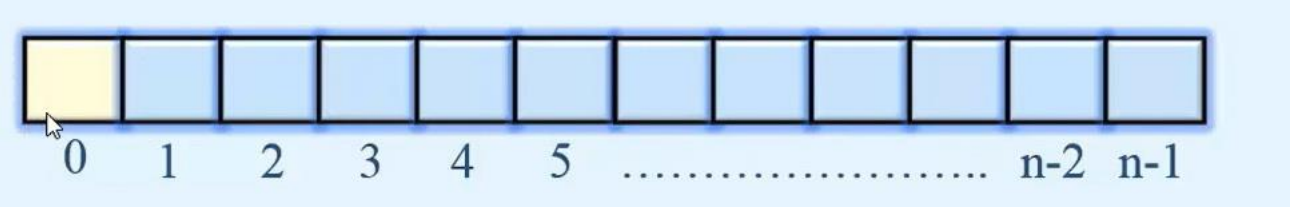
Niz x pre sortiranja

- Prolazak 1
  - Pronađi najmanji od elemenata x[0], x[1],....,x[n-1]
  - Pronađeni element razmeni sa x[0]
  - Posle prvog prolaska x[0] će sadržati najmanju vrednost u nizu



Niz x posle prvog prolaska

Prolazak 2

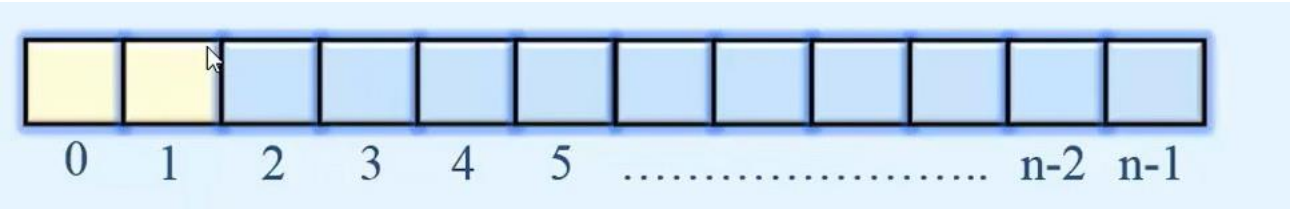


Niz x pre drugog prolaska

- Pronađi najmanji od elemenata x[1], x[2],....,x[n-1]
- Pronađeni element razmeni sa x[1]
- Posle drugog prolaska x[1] će sadržati drugu najmanju vrednost u nizu

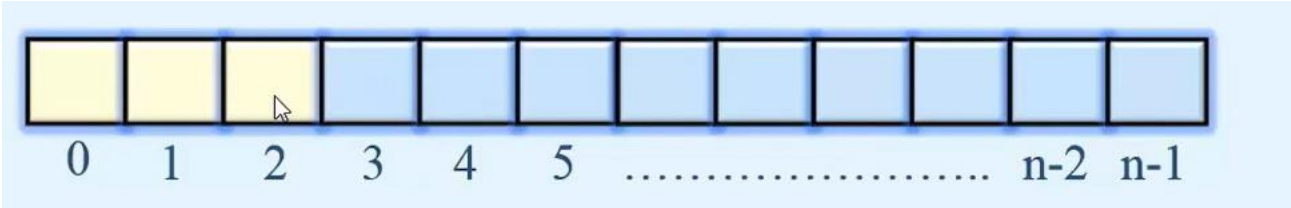
Niz x posle drugog prolaska

Prolazak 3



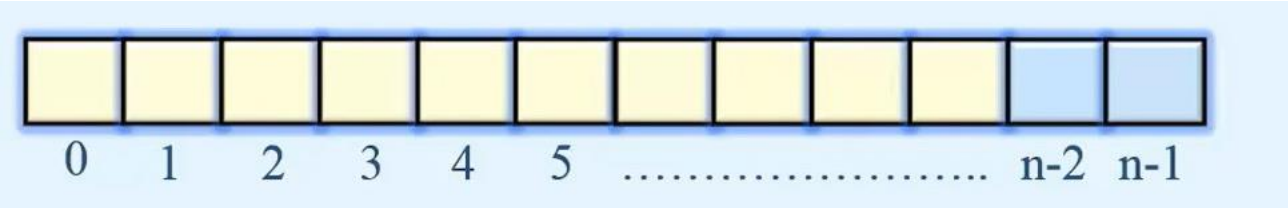
Niz x pre trećeg prolaska

- Pronađi najmanji od elemenata x[2], x[3],....,x[n-1]
- Pronađeni element razmeni sa x[2]
- Posle trećeg prolaska x[2] će sadržati treću najmanju vrednost u nizu



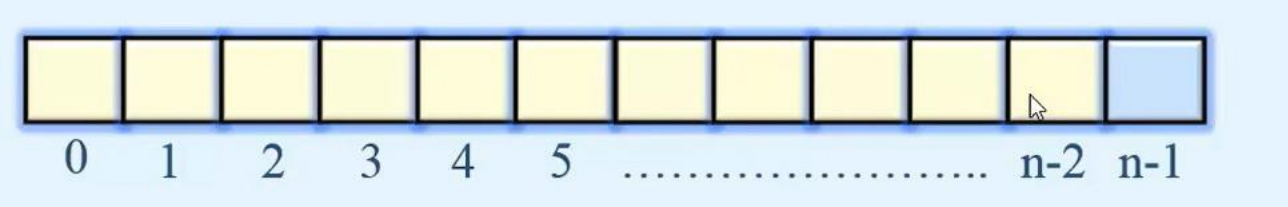
Niz x posle trećeg prolaska

Prolazak n-1



Niz x pre prolaska n-1

- Nađi najmanji od elemenata x[n-2] i x[n-1]
- Pronađeni element razmeni sa x[n-2]
- Posle prolaska n-1 niz će biti sortiran



Niz x posle prolaska n-1
Niz je sortiran

Algoritam Selection Sort

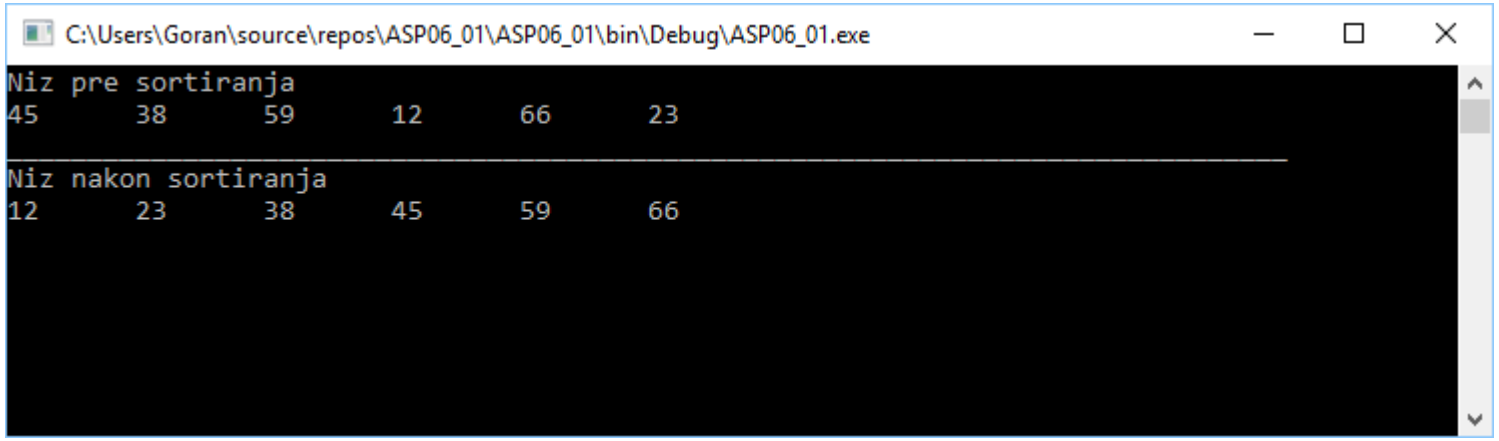
```
static void SelectionSort(int[] x)
{
    int n = x.Length;
    int temp = 0;
    int minIndex = 0;
    for (int i = 0; i < n - 1; i++)
    {
        minIndex = i;
        for (int j = i + 1; j < n; j++)
        {
            if (x[j]<x[minIndex])
            {
                minIndex = j;
            }
        }

        if (i != minIndex)
        {
            temp = x[i];
            x[i] = x[minIndex];
            x[minIndex] = temp;
        }

        //Console.Write($"Prolazak: {i + 1}\t ");
        //PisiNiz(x);
        //Linija(80);
     }
}
```

Pomoćne funkcije

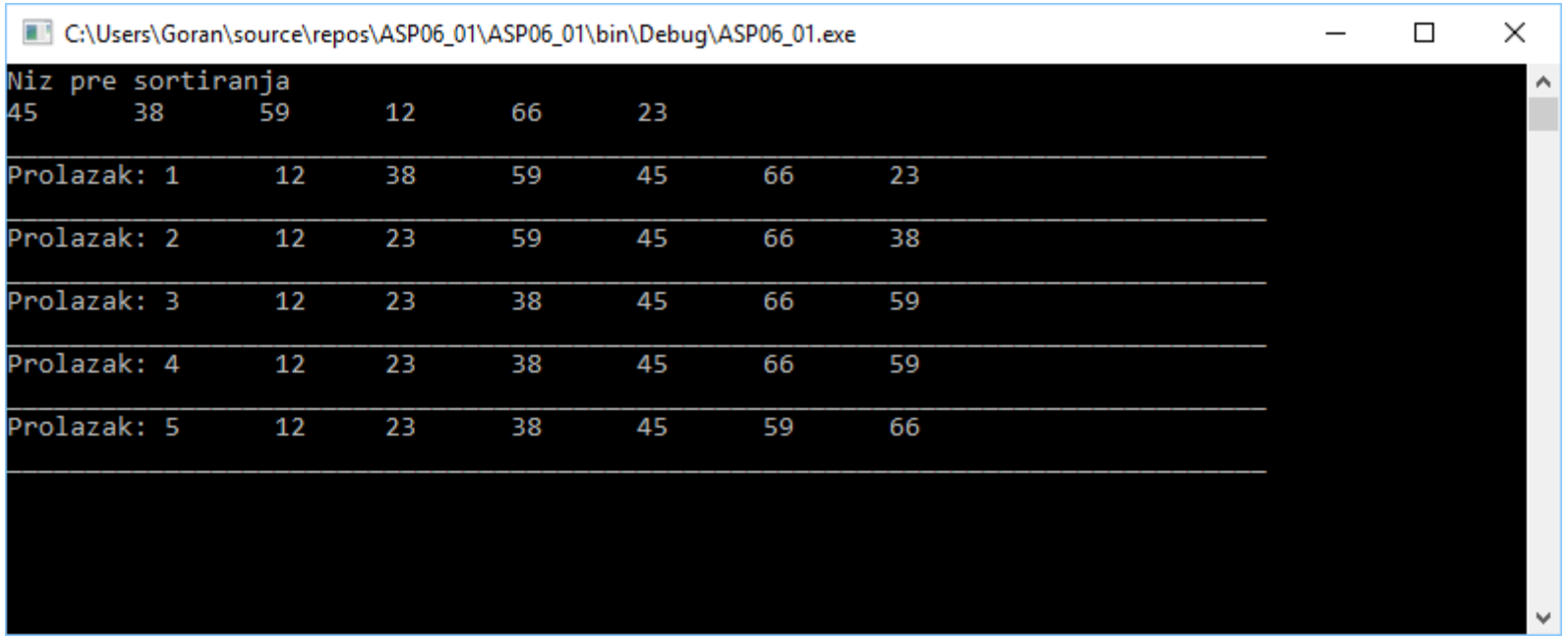
```
static int[] KreirajNiz(int n)
{
    Random rnd = new Random();
    int[] x = new int[n];
    for (int i = 0; i < n; i++)
    {
        x[i] = rnd.Next(1, 101); // od 1 do 100
    }
    return x;
}
static void PisiNiz(int[] x)
{
    for (int i = 0; i < x.Length; i++)
    {
        Console.Write(x[i] + "\t");
    }
    Console.WriteLine();
}

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

Poziv algoritma za sortiranje

```
static void Main(string[] args)
{
    int[] x = KreirajNiz(10);
    Console.WriteLine("Niz pre sortiranja");
    PisiNiz(x);
    Linija(80);
    SelectionSort(x);
    Console.WriteLine("Niz nakon sortiranja");
    PisiNiz(x);
    Console.ReadLine();
}
```



Prikaz niza na kraju svakog prolaska



Analiza Selection Sort algoritma -1

- Prolazak 1:
  - x[0] se upoređuje sa x[1], x[2],....,x[n-1]
  - ukupno (n-1) upoređivanja
- Prolazak 2:
  - x[1] se upoređuje sa x[2], x[3],....,x[n-1]
  - ukupno (n-2) upoređivanja
- Prolazak n-2:
  - x[n-3] se upoređuje sa x[n-2],x[n-1]
  - ukupno 2 upoređivanja
- Prolazak n-1:
  - x[n-2] se upoređuje sa x[n-1]
  - ukupno 1 upoređivanje

Analiza Selection Sort algoritma -2

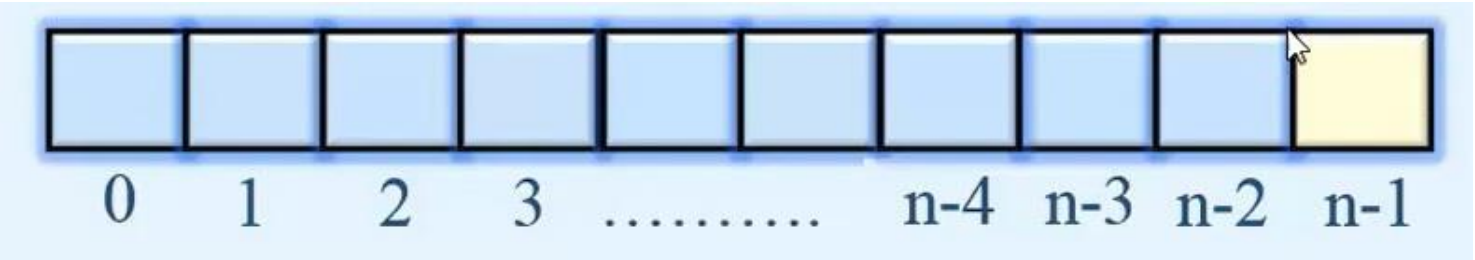
- Ukupno upoređivanja: $1 + 2 + \cdots + (n-1) = \frac{n}{2}(n-1)$
- Vremenska kompleksnost algoritma: $O(n^2)$
- Sortiranje nije osetljivo na podatke koji se sortiraju, podaci mogu biti sortirani u rastućem, opadajućem ili imati proizvoljan redosled
- U jednom prolasku samo jedna razmena
  - malo pomeranje podataka

Algoritam Bubble Sort

- Susedni elementi se upoređuju i razmenjuju im se mesta ukoliko redosled nije dobar
- Posle prvog prolaska najveći element niza dolazi na poslednju poziciju

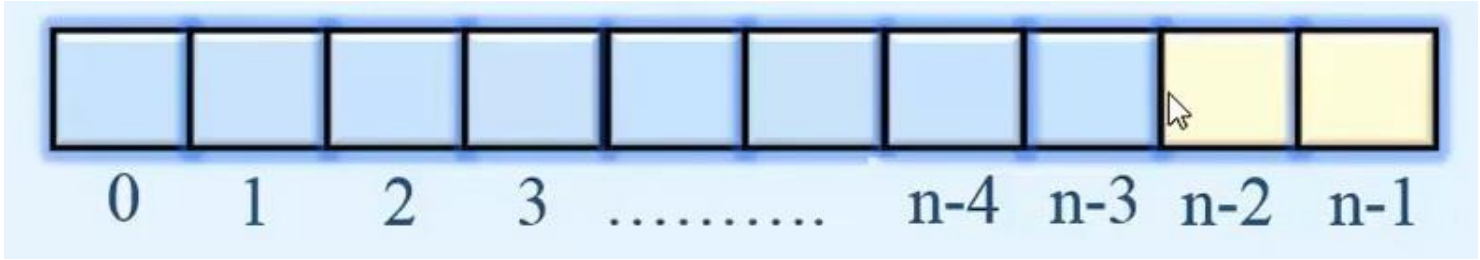
Prvi prolazak - Bubble Sort

- Upoređuje se x[0] i x[1] ako je x[0] > x[1] razmeni im mesta
- Upoređuje se x[1] i x[2] ako je x[1] > x[2] razmeni im mesta
- Upoređuje se x[n-3] i x[n-2] ako je x[n-3] > x[n-2] razmeni im mesta
- .....
- Upoređuje se x[n-2] i x[n-1] ako je x[n-2] > x[n-1] razmeni im mesta



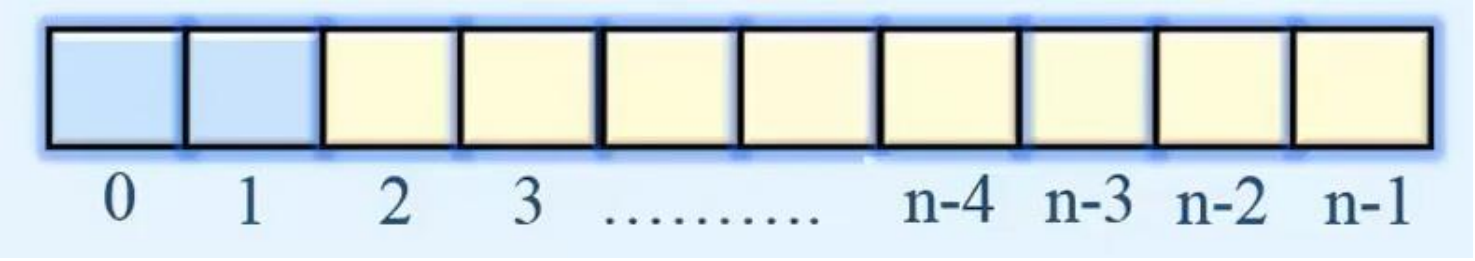
Drugi prolazak - Bubble Sort

- Upoređuje se x[0] i x[1] ako je x[0] > x[1] razmeni im mesta
- Upoređuje se x[1] i x[2] ako je x[1] > x[2] razmeni im mesta
- Upoređuje se x[n-4] i x[n-3] ako je x[n-4] > x[n-3] razmeni im mesta
- .....
- Upoređuje se x[n-3] i x[n-2] ako je x[n-3] >x[n-2] razmeni im mesta



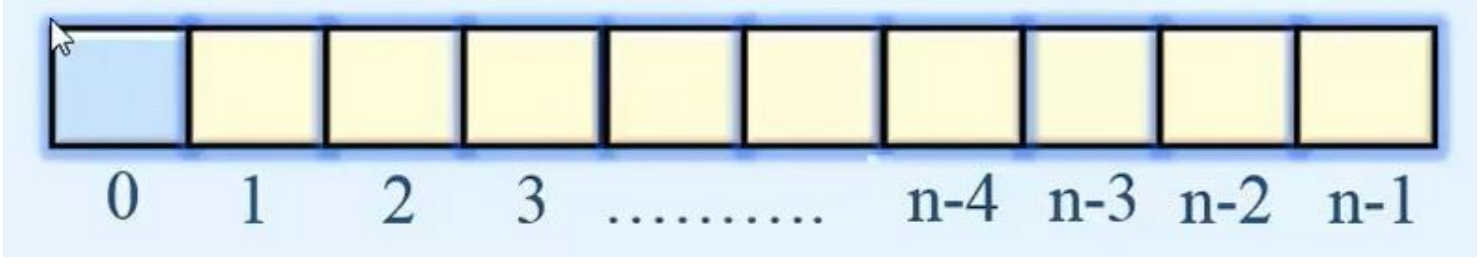
Prolazak n-1

Pre prolaska n-1

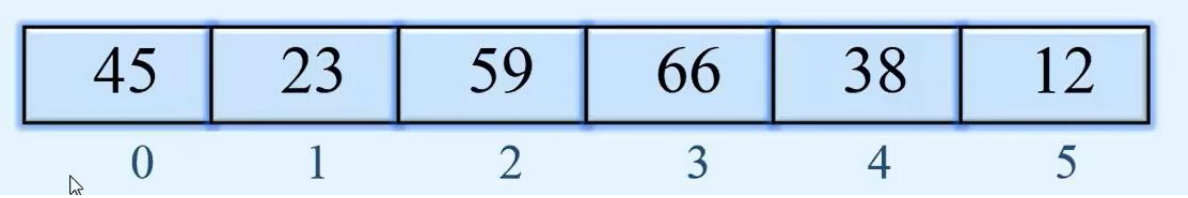


- Upoređuje se x[0] i x[1] ako je x[0] > x[1] razmeni im mesta
- Samo jedno poređenje

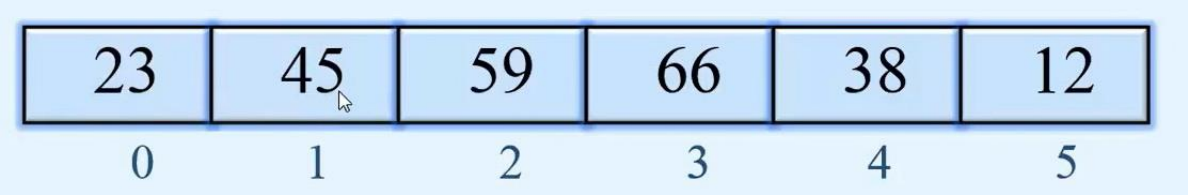
Posle prolaska n-1



Primer Bubble Sort prolazak 1



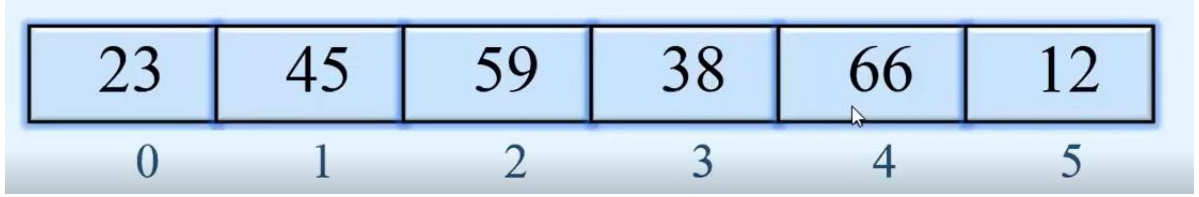
45 > 23 razmeni im mesta



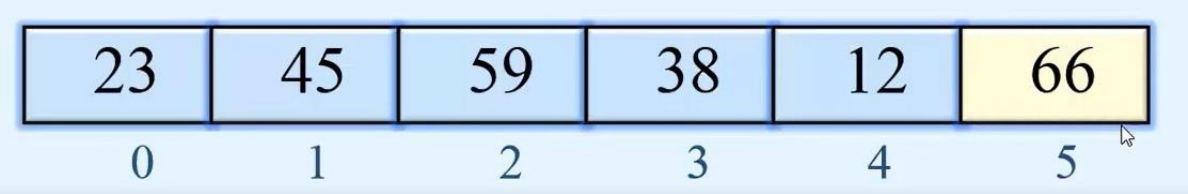
45 < 59, nema razmene

59 < 66, nema razmene

66 > 38, razmeni im mesta



66 > 12, razmeni im mesta



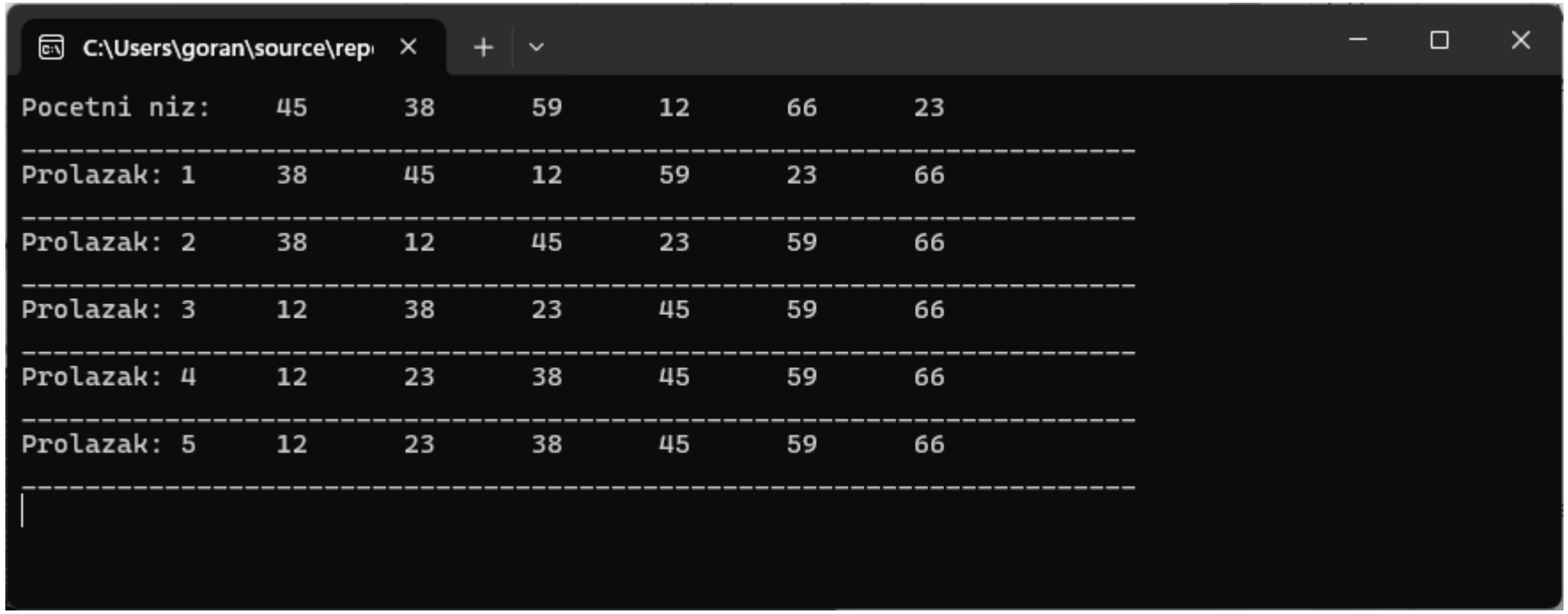
Primer n=6, sortiranje niza od 6 brojeva

Prolazak	uspoređuje se **x[j]** i **x[j + 1]**						Granice za j
i=0	0,1	1,2	2,3	3,4	4,5	j=0,1,2,3,4	0: 6-0-2
i=1	0,1	1,2	2,3	3,4		j=0,1,2,3	0: 6-1-2
i=2	0,1	1,2	2,3			j=0,1,2	0: 6-2-2
i=3	0,1	1,2				j=0,1	0: 6-3-2
i=4	0,1					j=0	0: 6-4-2
							0: n-i-2

Implementacija algoritma

```
static void BubbleSort(int[] x)
{
    int n = x.Length;
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            // Uporedi susedne elemenate
            if (x[j] > x[j + 1])
            {
                // Zamena ako nisu u ispravnom redosledu
                int temp = x[j];
                x[j] = x[j + 1];
                x[j + 1] = temp;
            }
        }
        //Console.Write($"Prolazak: {i+1}\t");
        //PisiNiz (x);
        //Linija(70);
    }
}
```

Sortiranje primenom Bubble Sort algoritma



Poboljšani Bubble Sort algoritam

```
static void BubbleSortRazmena(int[] x)
{
    int n = x.Length;
    bool imaRazmena; // Dodana varijabla za praćenje razmena

    for (int i = 0; i < n - 1; i++)
    {
        imaRazmena = false; // Inicijalizujemo na false na pocetku svakog prolaza

        for (int j = 0; j < n - i - 1; j++)
        {
            // Uporedi susedne elemente
            if (x[j] > x[j + 1])
            {
                // Zamena ako nisu u ispravnom redosledu
                int temp = x[j];
                x[j] = x[j + 1];
                x[j + 1] = temp;

                imaRazmena = true; // Postavi na true ako je doslo do razmene
            }
        }

        // Ako nije bilo razmena, niz je vec sortiran
        if (!imaRazmena)
            break;

    }
}
```

Analiza Bubble Sort algoritma -1

- Ako je niz već sortiran
  - Samo jedan prolazak
  - Ukupno n-1 poređenja
  - 0 razmena
  - Vremenska kompleksnost O(n)
- Ako je niz sortiran u opadajućem poretku
  - n-1 prolazak
  - Poređenja: $(n-1) + (n-2) + \cdots . + 2 + 1 = (n-1)n/2$
  - Razmena: $(n-1)n/2$
  - Vremenska kompleksnost $O(n^2)$

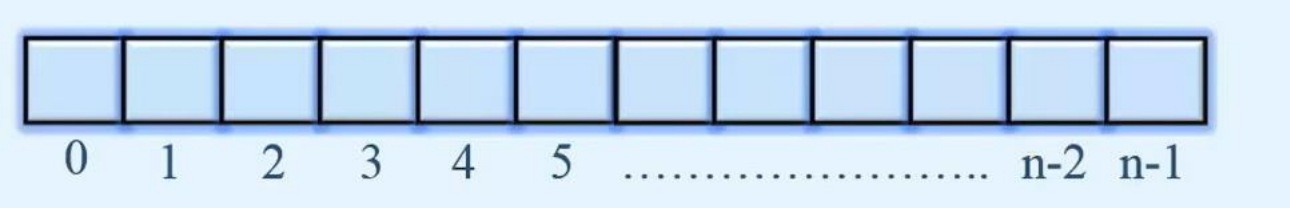
$1 + 2 + 3 + \cdots + n = n/2(n+1)$ - suma aritmetičkog niza

Analiza Bubble Sort algoritma -2

- Podaci su slučajno raspoređeni
- Spoljašnja petlja se izvršava $n - 1$ puta
- Unutrašnja petlja se izvršava $n - i - 1$
- Ukupan broj poređenja je suma aritmetičkog niza (n - 1) + (n - 2) + ... + 1, što je (n * (n - 1)) / 2.
- Vremenska kompleksnost $O(n^2)$

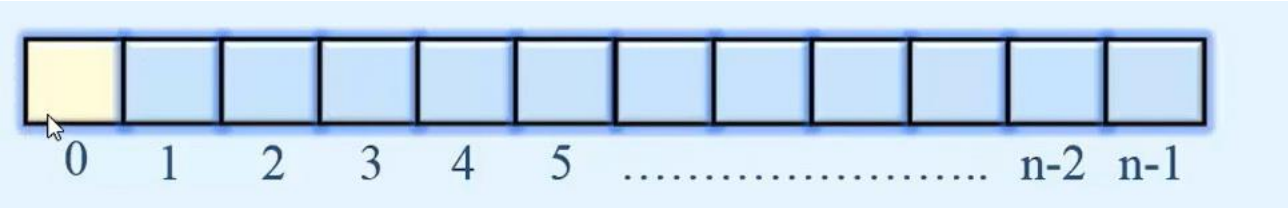
Algoritam Insertion Sort

- Niz x dužine n se postupno sortira u n-1 prolazaka. U svakom prolasku se produžava levi - sortirani dio niza za jedan element
- U i-tom prolasku se umeće element x[i] na njegovo pravo mesto između prvih $i - 1$ već uređenih elemenata u rastućem redosledu
- Element x[i] se smešta u promenljivu temp
  - temp = x[i]
  - Upoređuje se x[i-1] sa temp i ako je x[i-1]>temp, x[i-1] se pomera desno na poziciju x[i]
  - Upoređuje se x[i-2] sa temp i ako je x[i-2]>temp, x[i-2] se pomera desno na poziciju x[i-1]



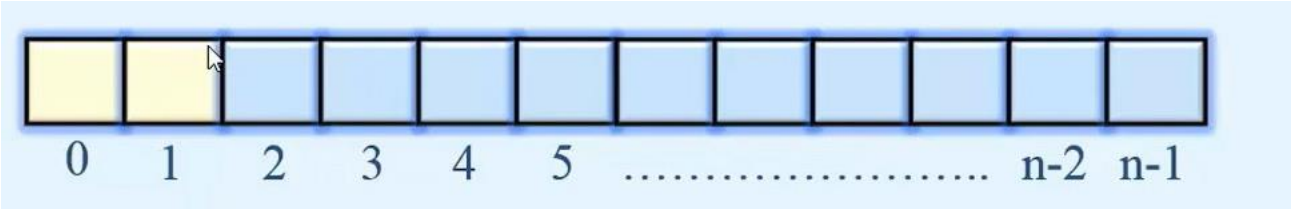
niz x

Prolazak 1

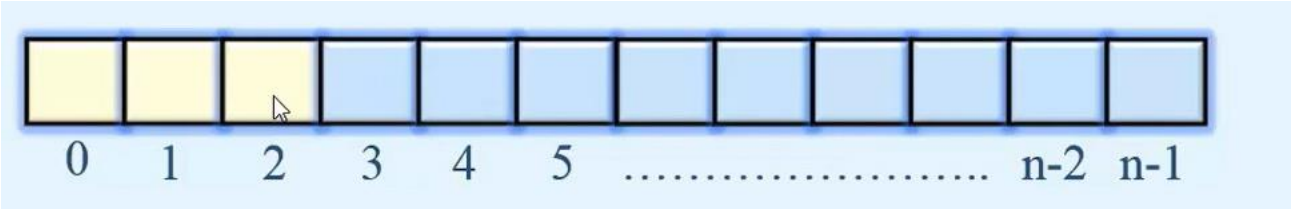


- Sortirani deo x[0]
- Nesortirani deo x[1],x[2],...x[n-1]
- Element x[1] se ubacuje u sortirani deo

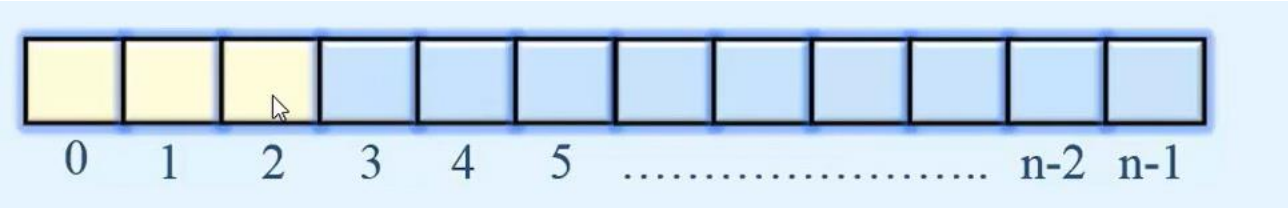
Prolazak 2



- Sortirani deo x[0],x[1]
- Nesortirani deo x[2],x[3],...,x[n-1]
- Element x[2] ubacuje se u sortirani deo na odgovarajuću poziciju

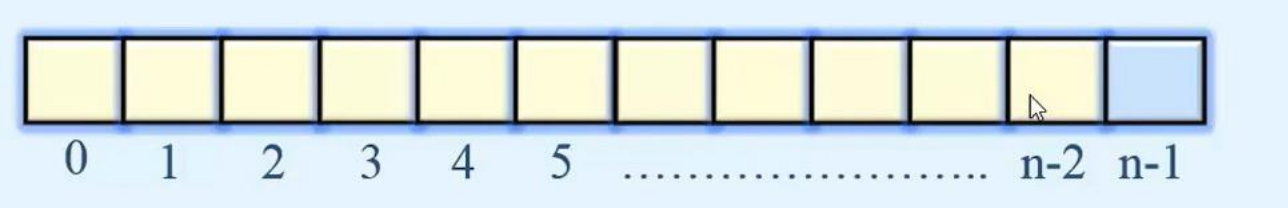


Prolazak 3



- Sortirani deo x[0],x[1],x[2]
- Nesortirani deo x[3],x[4],...,x[n-1]
- Element x[3] ubacuje se u sortirani deo na odgovarajuću poziciju

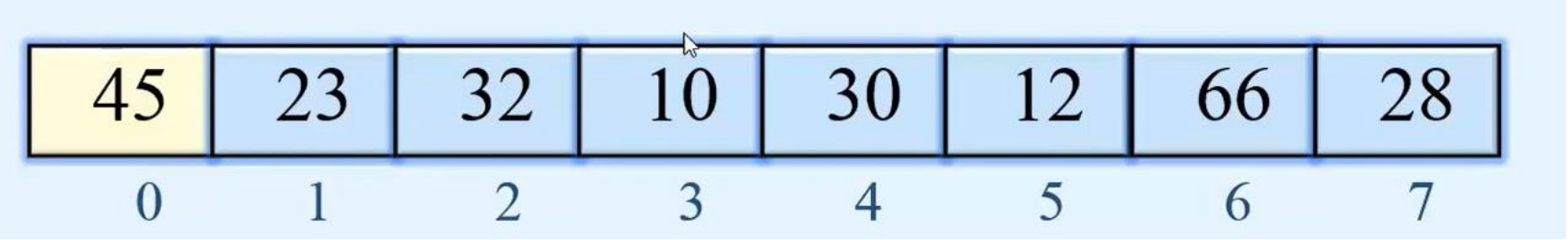
Prolazak n-1



- Sortirani deo x[0],x[1],...,x[n-2]
- Nesortirani deo x[n-1]
- Element x[n-1] ubacuje se u sortirani deo na odgovarajuću poziciju

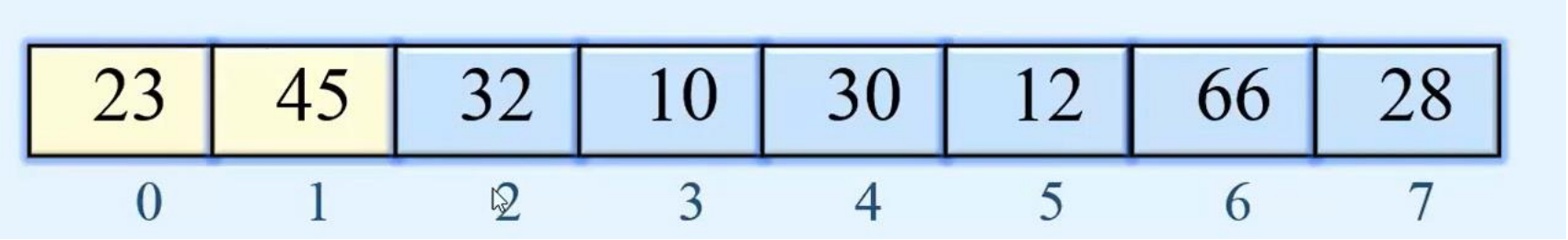
Niz je sortiran

Primer rada Insertion Sort algoritma – Prvi prolazak

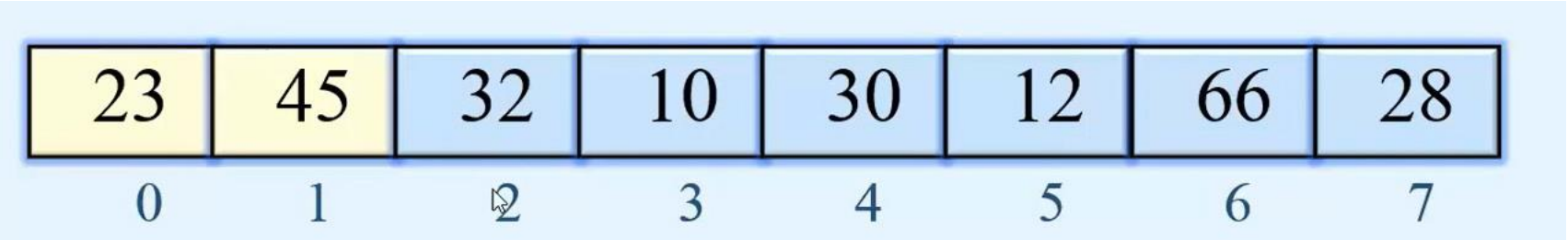


temp = 23

x[0]=45 >temp
45 pomeram desno, x[1] =45
na staroj poziciji broja 45 upisujem vrednost smeštenu u temp, x[0] =temp

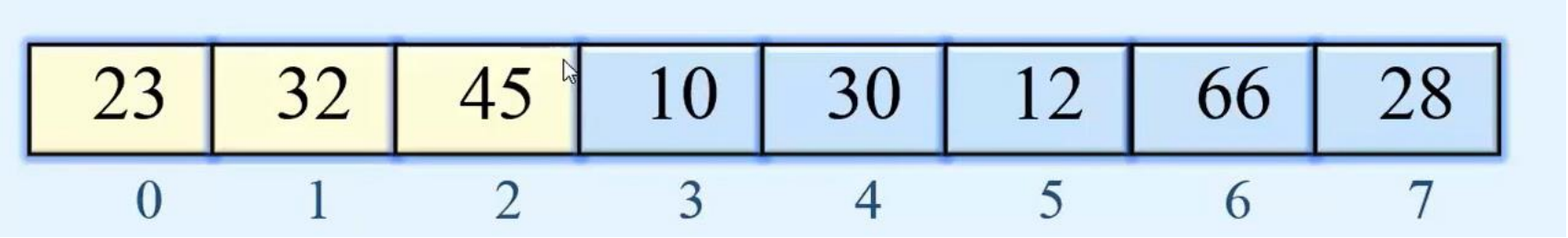


Drugi prolazak

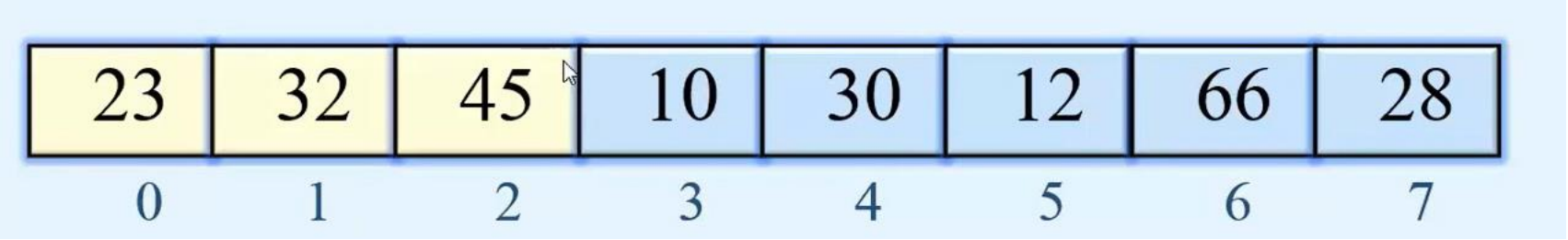


temp = 32

x[1]=45 >temp
45 pomeram desno, x[2] =45
x[0]=23 <temp, ne pomeram ga desno
na staroj poziciji broja 45 (koji je poslednji pomeren u desno)
upisujem vrednost smeštenu u temp, x[1] =32



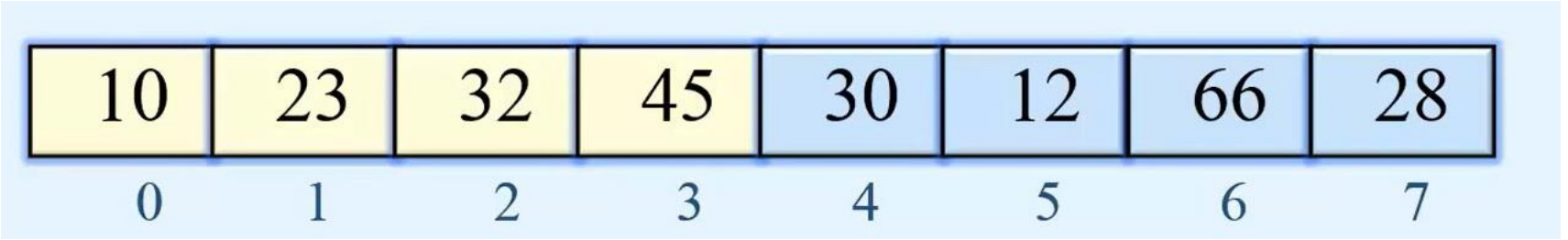
Treći prolazak



temp = 10

x[2]=45 >temp
45 pomeram desno, x[3] =45
x[1] = 32 > temp
32 pomeram desno x[2] =32
x[0]=23 >temp pomeram desno x[1]=23

na staroj poziciji broja 23 (koji je poslednji pomeren u desno) upisujem vrednost smeštenu u temp, x[0] =10

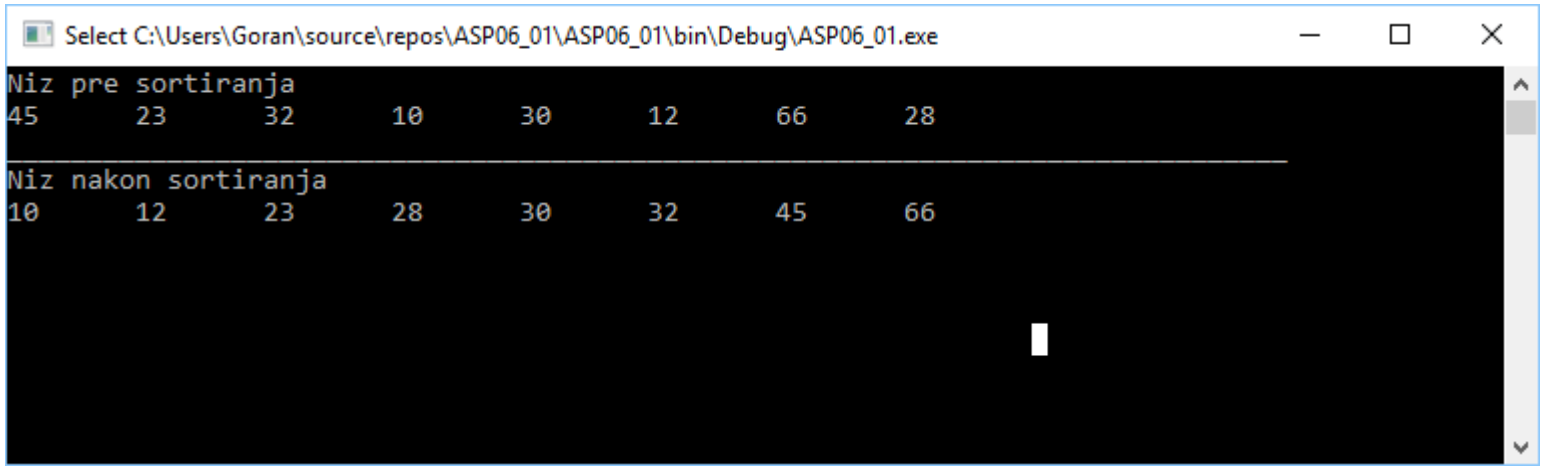


Implementacija algoritma

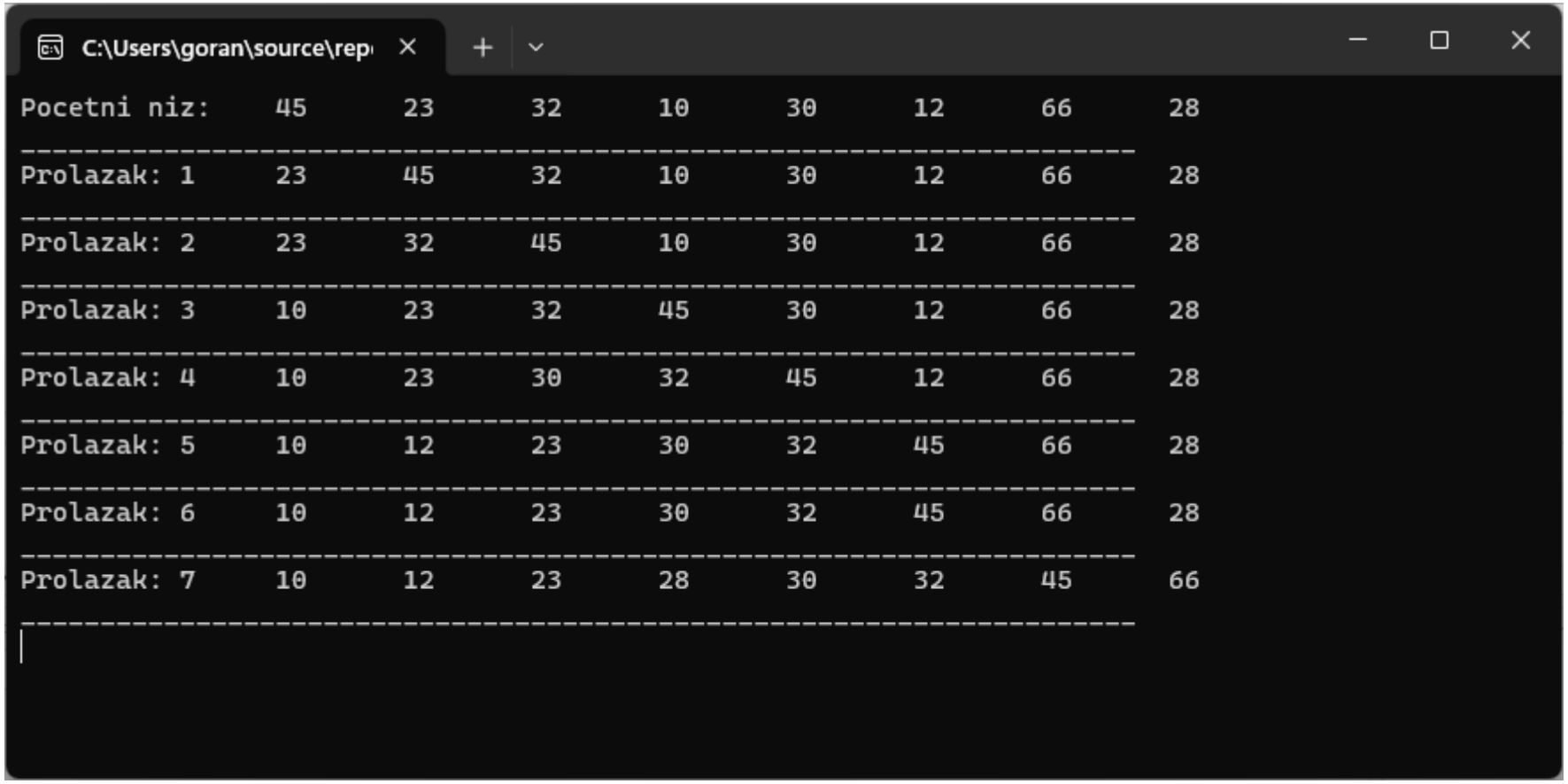
```
static void InsertionSort(int[] x)
{
    int n = x.Length;

    for (int i = 1; i < n; i++)
    {
        int temp = x[i];
        int j = i - 1;

        // pomeri elemente koji su veći od temp udesno
        while (j >= 0 && x[j] > temp)
        {
            x[j + 1] = x[j];
            j--;
        }

        // Ubaci temp na odgovarajuce mesto
        x[j + 1] = temp;
        Console.Write($"Prolazak: {i}\t");
        PisiNiz(x);
        Linija(70);
    }
}
```

Poziv algoritma

```
static void Main(string[] args)
{
    int[] x = { 45, 23, 32, 10,30,12, 66, 28 };
    Console.WriteLine("Niz pre sortiranja");
    PisiNiz(x);
    Linija(80);
    InsertionSort(x);
    Console.WriteLine("Niz nakon sortiranja");
    PisiNiz(x);
    Console.ReadLine();
}
```



Izvršavanje algoritma po koracima



Karakteristike Insertion Sort algoritma

- Algoritam započinje od drugog elementa u nizu (indeks 1), pa spoljašnja petlja ide od i=1 do i=n−1.
- Za i = 1, unutrašnja petlja se izvršava najviše 1 put
- Za i=n-1 unutrašnja petlja se izvršava najviše n-1 put
- Ukupan broj izvršavanja unutarnje petlje može se aproksimirati kao 1+2+3+…+(n−1), što je suma aritmetičkog niza, a to je n·(n−1)/2.
- Vremenska kompleksnost $O(n^2)$

Pitanje 1

Za algoritam Selection Sort važi sledeće tvrđenje:

a. Vremenska kompleksnost je najmanja kada su podaci sortirani u rastućem poretku
b. Vremenska kompleksnost je najmanja kada su podaci sortirani u rastućem poretku
c. Vremenska kompleksnost ne zavisi od podataka

Odgovor: c

Pitanje 2

Ako upoređujemo Bubble Sort i Selection Sort algoritam veći broj razmena vrednosti ima:

a. Bubble Sort
b. Selection Sort
c. Broj razmena je podjednak

Odgovor: a

Pitanje 3

Ako se radi sortiranje niza dužine n, algoritam Selection Sort ima sledeći broj prolazaka:

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

Odgovor: b

Pitanje 4

Kada se niz sortira korišćenjem Bubble Sort algoritma posle prvog prolaska:

a. Najveći element se nalazi na kraju niza
b. Najveći element se nalazi na početku niza
c. Najmanji element se nalazi na kraju niza

Odgovor: a

Pitanje 5

Kada se niz sortira korišćenjem Insertion Sort algoritma posle svakog prolaska:

a. Prvi element sortiranog dela ubacuje se u nesortirani deo
b. Prvi element nesortiranog dela ubacuje se u sortirani deo
c. Poslednji element nesortiranog dela ubacuje se u sortirani deo

Odgovor: b