Линеарна претрага низа

Пример: Пронаћи тражени елемент у низу целобројних вредности употребом линеарне претраге низа.

```
using System;
namespace Proba
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Uneti broj elemenata niza: ");
            int brojElemenataNiza =
                    Convert.ToInt32(Console.ReadLine());
            int[] niz = new int[brojElemenataNiza];
            Console.WriteLine("Uneti celobrojne vrednosti za elemente
                    niza: ");
            for(int i = 0; i < brojElemenataNiza; i++)
            {
                Console.Write($"niz[{i}] = ");
                niz[i] = Convert.ToInt32(Console.ReadLine());
            }
            Console.Write("Uneti celobroju vrednost koja se trazi u
                    nizu: ");
            int brojTrazen = Convert.ToInt32(Console.ReadLine());
            int nadjen = 0;
            for(int i = 0; i < brojElemenataNiza; i++)
            {
                if(niz[i] == brojTrazen)
                {
                    nadjen++;
                    Console.WriteLine($"Vrednost {brojTrazen} ima {i +
                            1}. element niza.");
                }
            }
            if (nadjen == 0)
            {
                Console.WriteLine($"\nVrednost {brojTrazen} nije
                        nadjena u nizu.");
            }
        }
    }
}
```

Даје:
Uneti broj elemenata niza: 4
Uneti celobrojne vrednosti za elemente niza:
niz[0] = 10
niz[1] = 20
niz[2] = 30
niz[3] = 40
Uneti celobroju vrednost koja se trazi u nizu: 40
Vrednost 40 ima 4. element niza.

Уносе се елементи низа у дефинисане меморијске локације којима припада низ.
Уноси се вредност која ће се претраживати да ли постоји у низу и поставља се иницијална вредност за промењиву nadjen на 0, што указује да у том моменту тражена вредност није пронађена у низу.
Петља пролази кроз цео низ и за сваки елемент који има тражену вредност приказује његов индекс у низу и поставља вредност промењиве nadjen на 1.
Поступак се понавља све док се за дужину низа уноси валидан број.
Метода линеарног претраживања се може користити код било које врсте низова (сортирани, несортирани) али пошто има велики број пролаза кроз низ (мора да испита сваки елемент низа) није погодна за низове изузетно великих дужина.

Пример: Одређивање индекса и вредности највећег елемента у низу реалних бројева

```
using System;
namespace Proba
{
    internal class Program
    {
        static void Main(string[] args)
        {
            const int MAX = 5;
            double[] niz = new double[MAX];
            Console.WriteLine("Uneti realne vrednosti za elemente
                    niza: ");
            for(int i = 0; i < MAX; i++)
            {
                Console.Write($"niz[{i}] = ");
                niz[i] = Convert.ToDouble(Console.ReadLine());
            }
            int imax = 0;
            for(int i = 0; i < MAX; i++)
            {
                if (niz[i] > niz[imax])
                {
                    imax = i;
                }
```

```
            }
            Console.WriteLine($"Najvecu vrednost {niz[imax]} ima {imax
                    + 1}. element niza.");
        }
    }
}
```

Даје:
Uneti realne vrednosti za elemente niza:
niz[0] = 5.34
niz[1] = 69.78
niz[2] = 11.34
niz[3] = -0.34
niz[4] = 7.56
Najvecu vrednost 69.78 ima 2. element niza.

На почетку кода се претпоставља да је у том моменту највећа вредност у низу као 1.елемент низа. После поређења тог елемента са следећим у низу, помоћу условне структуре се открива нова највећа вредност у низу и промењива imax добија индекс тог елемента у низу.

Пример: Сортирати низ: niz = {5, 2, 9, 4} методом уметања елемената

```
using System;
namespace Proba
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int[] niz = { 5, 2, 9, 4 };
            int pom, i, j;
            for(i = 0; i < 4; i++)
            {
                pom = niz[i];
                for(j = i - 1; j >= 0; j--)
                {
                    if (niz[j] > pom)
                    {
                        niz[j + 1] = niz[j];
                    }
                    else
                    {
                        break;
                    }
                }
                niz[j + 1] = pom;
```

```
            }
            Console.WriteLine("Sortiran niz: ");
            for(i = 0; i < 4; i++)
            {
                Console.Write(niz[i] + " ");
            }
            Console.WriteLine();
        }
    }
}
```

Ако користи методу уметања, програм у помоћној промењивој памти редом сваки елемент (од другог до последњег) и испитује релацију у којој је овај елемент са претходним елементима.
Све док релација није очекивана, помера претходне елементе удесно, кад то више није случај умеће упамћену вредност на своје место.