Низовни тип

Научили смо да саберемо два броја као две променљиве одређеног типа или да оперишемо са неколико вредности.Замислите сада да пишете програм који врши обраду великог броја података.Велики број података би захтевао постојање исто тако великог броја променљивих да би сместити њихове вредности, што никако није практично.

Слика 10.1. Гомиле

Како би се онда снашли са таквом гомилом ?

За решавање оваквих проблема су погодни низовски типови података, које смо раније споменули у оквиру 4.теме нашег курса. Сада ћемо се детаљније упознати са њима.

Дефиниција: Низ је колекција сличних елемената,а редни број одређеног елемента се назива ИНДЕКС

Да бисмо у програм унели онолико података колико желимо, користећи само једну променљиву и да би их једноставније учитали и обрадили, потребни су нам НИЗОВИ.Углавном ћемо се бавити једнодимензионим низовима.

Низови се у C# декларишу, тако што се испред имена променљиве ставе угласте заграде.

```
//Tip niza brojevi je int
int [] brojevi ;
```

Низ тек треба да се дефинише (креира). Низовска променљива brojevi је само локација у меморији,која може да чува адресу низа,што ће бити објашњено касније. За креирање самог низа мора се задати његов тип и колико ће елемената садржати.

Да би дефинисали низ користимо кључну реч **new**:

```
//Naredba kreira niz koji ce sadrzati 5 clanova tipa int
int [] brojevi = newint[5] ;
```

На овај начин смо резервисали део меморије за 5 узастопних елемената нашег низа.Још треба унети те елементе!

Почетне вредности елемената (које се касније могу мењати по потреби) можемо унети на разне начине.Можемо извршити директну доделу вредности што је непрактично за огромне низове.

```
//Navodjenje elemenata
int [] brojevi = {1,2,3,4,5};
```

или

```
int [] brojevi = newint[5] ;
//Svakom elementu niza brojevi dodeljujemo vrednost pojedinacno
brojevi[0]=1;
```

```
brojevi[1]=2;
......
brojevi[4]=5;
```

За пролажење кроз низ користимо неку од петљи, нпр. for, па на овај начин најчешће и уносимо елементе низа.

```
int [] brojevi = new int[5];

//Niz brojevi popunjavamo redom vrednostima 1,2,3,4,5
for (int i = 0; i < 5; i++)
{
//Kako brojac krece od nule,njegovu vrednost svaki put uvecavamo za 1
brojevi[i] = i+1;

}
```

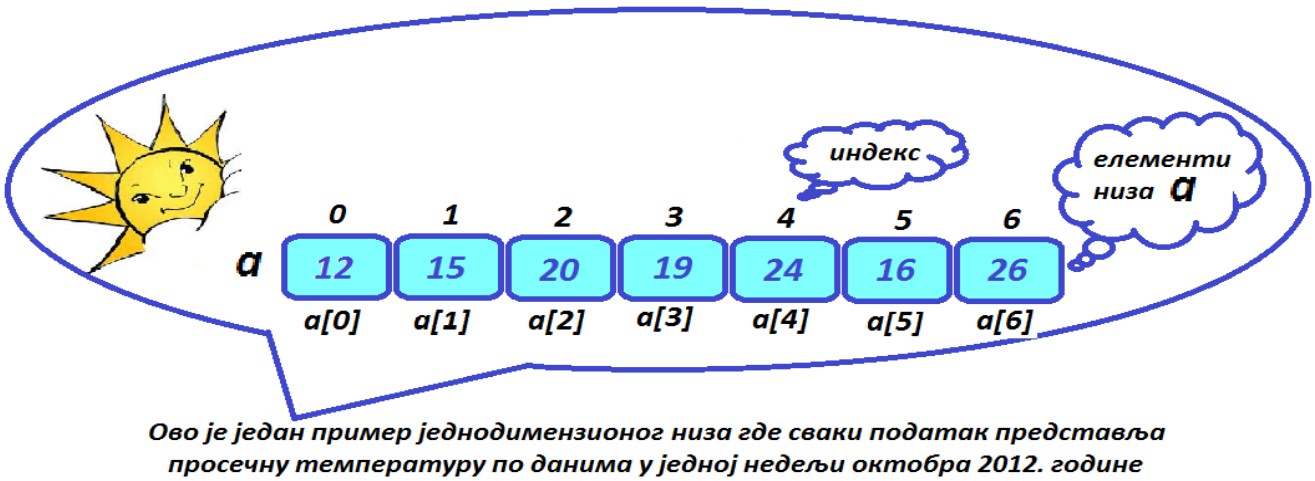
Можемо искористити for петљу и на овај начин, ако хоћемо да корисник уноси елементе са стандардног улаза.

```
//Definisemo prostor u memoriji za najvise 5 elemenata niza brojevi
int[] brojevi = new int[5];

Console.WriteLine("Unesi elemente niza:";);
//Unosimo clanove niza sa tastature, pri cemu se ucitava element po element
for (int i = 0; i < 5; i++)
{
brojevi[i] = int.Parse(Console.Readline());
}
```

- **ТИП**низа може бити било који тип: *int,float,double,string,...*
- Индекс низа мора бити цео број и њиме је једнозначно одређен елемент који се налази на том редном броју!
- Ако је *brojevi[3] = 5;*индекс је 3, што значи да је на четвртом месту у низу *brojevi* елемент вредности 5.
- Индекси се увек крећу од ***0*** до ***n-1***, где је ***n*** произвољан број елемената низа.
- Ако случајно покушамо да приступимо елементу чији је индекс ван овог интервала програм ће избацити грешку!

```
int[] brojevi = new int[5];
//Prvi element niza od 5 elemenata
brojevi[0] = 2;
//Poslednji element niza od 5 elemenata
brojevi[4] = 5;
```

- Низови могу бити: једнодимензиони, дводимензиони и вишедимензиони.



Слика 10.2. Температуре

Ево још неких примера дефинисања једнодимензионих низова:

```
//Promenljiva niz sadrzi najvise 50 elemenata realnog tipa double
double [] niz = new double[50];
//Osmi element niza je 12 tj.na 8. mestu je vrednost 12
niz[7] = 12;
//Niz karaktera
char[] slova = {'a','b','c','d','h'};
//Niz stringova
string [] imena= {'Mira','Jovana','Marko','Sreta'};
/*Promenljiva parni_niz ukazuje na 40 rezervisanih mesta u memoriji za elemente
tipa int*/
int parni_niz[40];
```

Основне карактеристике низова

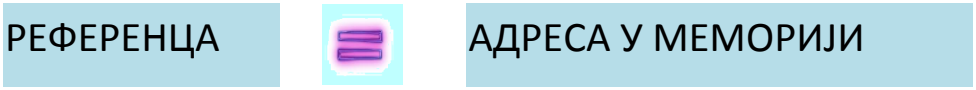
Приступ елементима низа

Да бисмо прочитали или променили вредност неког елемента, можемо му приступити навођењем индекса тог елемента у угластим заградама иза одговарајуће промењиве низовног типа.

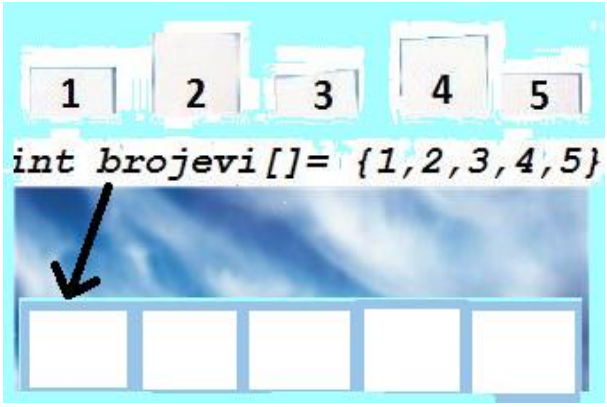
```
//Promenljiva p dobija vrednost petog elementa niza a
int p = a[4];
//Sedmi element niza a dobija vrednost 25
a[6]=25;
//Ovom naredbom ispisujemo element a[5] u konzoli
Console.WriteLine(a[5]);
int [] b = new int[200];
//Ucitava se vrednost i smesta u 58.clan niza "b"
Console.ReadLine (b[57]);
```

Низовска променљива

Покушаћемо да објаснимо укратко зашто је ово тако посебна променљива и какве она има везе са референцама.Поменули смо раније да је низовска променљива само локација у меморији, да не садржи сам низ и да може да чува адресу првог елемента низа.Ако задамо број елемената низа, резервисали смо део меморије и променљива низовског типа ће референсати (упућивати) на ту локацију у меморији.Што значи да је ова променљива одвојена од низа на који реферише.



Следећа сличица показује да низовска променљива brojevi чува адресу почетног елемента у меморији, а не саме вредности елемената.А чим знамо позицију првог елемента лако можемо приступити и осталим елементима.



Слика 10.3. Низови у меморији

Дужина низа

- То је број елемената које низ садржи.
- Атрибут *.length* враћа број елемената низа. На пример b.length за наш претходни низ имаће вредност 200.
- Овај атрибут се може користити за контролисање *for* петље која обрађује елементе низа.


```
//brojevi.Length ima vrednost 7, tako da ce brojac ici do 7
for (int i = 0; i < brojevi.Length; i++)
```

У програму је некада потребно проверити да ли одређени низ садржи елементе, да не би дошло до избацивања изузетка.Ако нема елемената, програм може исписати поруку да је низ празан.

```
int[] brojevi = new int[5];
```

```
if(brojevi.Length == 0)
//Program ce ispisati poruku u konzoli
MessageBox.Show("Niz je prazan");
```

Решавамо неке краће примере:

Пример 1. Формирати низ који има 15 чланова.Члановима са парним индексима доделити вредност 1, а члановима са непарним индексима доделити вредност 0.Исписати чланове добијеног низа.

Решење:

```
//Najpre deklarisemo potrebne promenljive
int i;
int[] a = new int[15];
//Unosimo elemente niza
for (i = 0; i < 15; i++)
{
if(i % 2 == 0)
//Ako je indeks paran
a[i] = 1;
else
//Ako je indeks neparan
a[i] = 0;
}

//Ispisujemo clanove niza u konzoli po pokretanju programa

for(i = 0;i < 15;i++)

Console.WriteLine("Elementi formiranog niza su:"+a[i]);
}
```

Пример 2. Дат је низ a који садржи температуре свих дана у једној недељи.Написати програм који ће заменити температуре сваког петог и шестог дана у датом низу.Исписати новодобијени низ.

Решење:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
classProgram
    {
static void Main(string[] args)
        {
//Deklaracija promenljivih
int [] a = new int[7];
int i;
//Pomocna promenljiva za zamenu vrednosti
int p;
```

```
//Unosimo elemente niza
Console.WriteLine("Unesi elemente niza:");
for(i = 0;i < 7; i++)
            {
               /*Metod int.Parse konvertuje string promenljivu u numericku vrednost
tipa int.
               Unete cifre se inicijalno posmatraju kao stringovi.*/
a[i] = int.Parse(Console.ReadLine());
            }
//Zamenjujemo mesta petog i sestog elementa u nizu
            p = a[4];
a[4] = a[5];
a[5] = p;
//Ispisujemo clanove novoformiranog niza
for (i = 0; i < 7; i++)
Console.WriteLine("Elementi su:" + a[i]+" ");

        }

    }
}
```

Пример 3. Радник у магацину жели да напише програм који ће му помоћи да ради са залихама робе у магацину.Он жели да са стандардног улаза унесе шифре производа, количину производа и цену, да одреди укупну вредност залиха и шифре оних производа чија вредност залиха прелази 1/10 укупне вредности залиха у магацину.

Решење:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
classProgram
    {
static void Main(string[] args)
        {
//Sifre, kolicinu i cene proizvoda smestamo u nizove
int[] sifra = new int[150];
       //Pretpostavimo da kolicine i cene proizvoda mogu biti decimalni brojevi, tipa
float
float[] kolicina = new float[150];
float[] cena = new float[150];

int i, unos;
```

```
//Ukupna vrednost proizvoda u skladistu
float vrednost = 0;

//Unosimo podatke sa standardnog ulaza(tastatura)
Console.WriteLine("Za koliko proizvoda zelite da unesete podatke");
unos = int.Parse(Console.ReadLine());

Console.WriteLine("Unesite sifre proizvoda");
for (i = 0; i < unos; i++)
            {
sifra[i]= int.Parse(Console.ReadLine());
            }

Console.WriteLine("Unesite kolicine proizvoda");
for (i = 0; i < unos; i++)
            {
                /*Svaki element niza kolicina je broj primeraka nekog proizvoda
                koji je stigao u magacin*/
kolicina[i] = float.Parse(Console.ReadLine());
            }

Console.WriteLine("Unesite cene proizvoda");
for (i = 0; i < unos; i++)
            {
cena[i] = float.Parse(Console.ReadLine());
            }

//Racunamo vrednost svih proizvoda u skladistu
for (i = 0; i < unos; i++)
            {
                /*Vrednost zalihe jedne vrste proizvoda u magacinu dobijamo kada
pomnozimo kolicinu
                tog proizvoda i cenu za svaki pojedinacan primerak. Ukupnu vrednost
dobijamo kada
                saberemo vrednosti svih vrsta proizvoda.*/
vrednost += kolicina[i] * cena[i];
            }
Console.WriteLine("Ukupna vrednost proizvoda u skladistu je {0}",vrednost);

/*Ispisujemo sifre proizvoda kojima je vrednost zaliha veca od 1/10 vrednosti
                        zaliha svih proizvoda*/

for (i = 0; i < unos; i++)
            {
if (cena[i] * kolicina[i] > vrednost/10)
                {
```

```
Console.WriteLine(" " +sifra[i]);
                }
            }
        }
    }
}
```

Пример 4. Власник ланца хотела жели да израчуна број гостију у сваком хотелу у односу на укупан број гостију у свим хотелима изражено у процентима.

Решење:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication7
{
classProgram
    {
        /*Argumenti funkcije procenat: promenljiva n se odnosi na broj hotela, a svaki
element
        niza a predstavlja broj gostiju u pojedinacnom hotelu*/
static void procenat(int[] a, int n)
        {
int i, j;
int suma=0;
for (i = 0; i < n; i++)
            {
//Racunamo ukupan broj gostiju u svim hotelima
suma+=a[i];
            }

//Ispisujemo koji se procenat ukupnog broja gostiju nalazi u svakom od hotela
for (j = 0; j < n; j++)
            {
Console.WriteLine("U",j+1,". hotelu je smesteno", a[j]/suma*100,"% od
                                ukupnog broja gostiju");
            }
        }
static void Main(string[] args)
        {
int[] hoteli =  new int[15];
int i;
int n;

Console.WriteLine("Unesite broj hotela za koje zelite da vrsite izracunavanje");
```

```
            n = int.Parse(Console.ReadLine());

Console.WriteLine("Unesite broj gostiju za svaki hotel posebno");
for (i = 0; i < n; i++)
            {
hoteli[i] = int.Parse(Console.ReadLine());
            }
       //Pozivamo funkciju procenat da nam izracuna odnos gostiju
procenat(hoteli,n);
        }
    }
}
```

Напомена: Када отворимо нови пројекат, код програма уносимо у прозору**Program.cs**у оквиру витичастих заграда**static void Main**метода.

Израчунавање просечне вредности

Ако желимо да пронађемо просек неког скупа елемената, биће нам потребна само једна *for* петља.Можемо унети и сабрати вредности у једној итерацији, а затим добијену суму поделити бројем елемената из дате колекције. Као што примећујете, низ нам није био потребан! Све смо решили додавањем једног по једног елемента формирајући суму. Ово је могуће све док се не јави проблем који захтева**два пролаза**, где је већ неопходно**искористити низ**! У другој итерацији би могли да тражимо елемент који је испод или изнад просечне вредности.

Погледајмо следећи пример:

Хоћемо да напишемо програм који израчунава колики је натпросечан резултат постигнут на Олимпијским играма у Лондону 2012.године.

Како знамо да на оваквим спортским дешавањима учествује огроман број спортиста из целога света, било би незгодно формирати низ који би садржао освојене поене сваког од њих, зато ћемо се задовољити неким произвољно великим бројем уноса, нпр.да одредимо просек за првих 50 такмичара или више, што ће бити могуће. Затим ћемо приказати све учеснике који су освојили већи број поена од просечне вредности.

Упутство: Елементе уносити са стандардног улаза све док се не унесе одређена вредност нпр. 0 или наредба за прекид *break.*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
classProgram
    {
static void Main(string[] args)
        {
//Deklarisanje promenljivih
int i;
double prosek;
//n evidentira broj unetih rezultata takmicara
int n = 0;
//b cuva tekucu unetu vrednost
int b;
int suma = 0;
```

```
int []a = newint[50];

Console.WriteLine("Unesi elemente niza:");

      //Sa standardnog ulaza se unosi broj poena prvog takmicara
            b = int.Parse(Console.ReadLine());

while(b != 0)
            {
            /*Poene svakog takmicara smestamo u niz a, dok se promenljivom n
izrazavapozicija takmicara u nizu*/
a[n] = b;
//Sve dok ne unesemo nulu, promenljiva n se uvecava
n++;
      //Unosimo poene sledeceg takmicara
                b = int.Parse(Console.ReadLine());
            }

//Prikaz tacnog broja unetih elemenata niza
Console.WriteLine("Uneli ste"+ n + "elemenata");

for(i = 0;i < n;i++)
            {
      //Sabiramo poene svih ucesnika takmicenja
suma = suma + a[i];
            }

      //Racunamo prosecnu vrednost na takmicenju
prosek = suma / n;

Console.WriteLine("Prosecan rezultat koji je postiglo"+ n +"takmicara je" + prosek);

      //Proveravamo da li ima ucesnika sa vecim brojem poena od prosecnog
for(i = 0;i < n;i++)
            {
if(a[i] > prosek)
                {
Console.WriteLine("Poeni takmicara koji su iznadprosecnog rezultata:" + a[i] + "/n");
                }
            }
      }
    }
}
```

Напомена: Да не би дошло до забуне, користили смо промењиву n као бројач у while петљи, пошто смо желели да она представља и тачан број унетих елемената у наставку задатка, где смо је у тој функцији и користили у for итерацији сабирања чланова низа.

Пример 2. Израчунати просечну висину ученика једног одељења као и висину ученика која је најближа просечној.

Решење:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
classProgram
    {
static void Main(string[] args)
```

```
        {
        int i,n;
double prosek;
        //Promenljiva najblizi ce cuvati visinu ucenika najblizu prosecnoj visini svih
ucenika
        int najblizi;
        int suma = 0;

//Pretpostavimo da odeljenje nema vise od 30 ucenika
double[] a = newdouble[30];

        //Sa standardnog ulaza unosimo broj ucenika
        Console.WriteLine("Unesi broj ucenika datog odeljenja:");

            n = int.Parse(Console.ReadLine());

        //Unosimo visine ucenika(mogu biti realnog tipa double) i smestamo ih u niz a[]
        Console.WriteLine("Unesi visine ucenika: ");

        for (int i = 0; i < n; i++)
            {
a[i] =Double.Parse(Console.ReadLine());
            }
//Racunamo sumu visina svih ucenika
for(i = 0;i < n;i++)
            {
suma = suma + a[i];
            }
prosek = suma / n;

//Postavljamo prvi elemenat niza da bude najblizi proseku
najblizi = a[0];
/*Kako smo pretpostavili da je visina pravog ucenika najbliza proseku,
        razmatramo visine pocev od drugog elemnta u nizu - i=1*/
for(i = 1;i < n;i++)
            {
//Math.Abs() vraca apsolutnu vrednost broja u zagradi
/*Ako je razlika izmedju tekuce visine i prosecne manja od razlike tekuce najblize
visine prosecnoj i prosecne, onda takuca visina a[i] postaje najbliza prosecnoj*/
if(Math.Abs(a[i]-prosek) <Math.Abs(najblizi-prosek))
                {
najblizi=a[i];
                }
            }
//Ispisujemo visinu ucenika koja je najbliza prosecnoj visini celog odeljenja
```

```
Console.WriteLine("Visina ucenika najbliza prosecnoj visini je:"+ najblizi);
        }
    }
}
```

Пример 3. Написати програм који учитава низ бројева типа double и рачуна аритметичку, геометријску и хармонијску средину унетих бројева.

Решење:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication2
{
classProgram
    {
//Definisemo funkciju koja racuna aritmeticku sredinu n clanova niza niz[]
static double aritmeticka_sredina( double[] niz, double n)
        {
int i;
double sredina, suma = 0;
for (i = 0; i < n; i++)
            {
//Racunamo sumu svih elemenata niza
suma+=niz[i];
            }

//Aritmeticka sredina je jednaka kolicniku sume svih clanova i broja clanova
sredina = suma / n;
return sredina;
        }
//Definisemo funkciju koja racuna geometrijsku sredinu n clanova niza niz[]
static double geometrijska_sredina( double[] niz,  double n)
        {
int i;
double sredina = 0.0;
double proizvod = 1;
for (i = 0; i < n; i++)
            {
//Racunamo proizvod svih clanova niza niz[]
proizvod *= niz[i];
            }

//1/n*exp(x) je n-ti koren iz x
//Math.Exp i Math.Log definisane funkcije u C#
```

```
sredina = Math.Exp(1/n*Math.Log(proizvod));
return sredina;
        }
//Definisemo funkciju koja racuna harmonijsku sredinu n clanova niza niz[]
static double harmonijska_sredina( double[] niz,  double n)
        {
int i;
double sredina;
       //Promenljiva rsuma predstavljace zbir reciprocnih vrednosti clanova niza niz[]
double rsuma = 0;

for (i = 0; i < n; i++)
            {
//rsuma = 1/niz[0] + 1/niz[1] +....+ 1/niz[n-1]
rsuma += (1 / niz[i]);
            }
       /*Harmonijsku sredinu dobijamo kao kolicnik broja elemenata niza i sume
       reciprocnih vrednosti elemenata*/
sredina = n / rsuma;
return sredina;
        }
static void Main(string[] args)
        {
double[] a = new double[50];
double dim;
int i;
double a_sredina, h_sredina, g_sredina;

Console.WriteLine("Unesite broj clanova niza");
dim = double.Parse(Console.ReadLine());

Console.WriteLine("Unesite clanove niza");
for (i = 0; i < dim; i++)
            {
a[i] = double.Parse(Console.ReadLine());
            }

/*Pozivamo funkcije koje ce nam izracunati aritmeticku, geometrijsku
i harmonijsku sredinu elemenata niza a*/
            a_sredina = aritmeticka_sredina(a,dim);
            g_sredina = geometrijska_sredina(a,dim);
            h_sredina = harmonijska_sredina(a, dim);

Console.WriteLine("Aritmeticka sredina unetih brojeva je:", a_sredina);
Console.WriteLine("Geometrijska sredina unetih brojeva je:", g_sredina);
Console.WriteLine("Harmonijska sredina unetih brojeva je:", h_sredina);
```

```
        }
    }
}
```

Сортирање низа

Дефиниција: Сортирање је процес уређења елемената низа у растућем или опадајућем поретку.

Размотримо сада једну помоћну фор петљу која ће нам помоћи да боље разумемо поступак сортирања низова! i-ти елемент у низу a заменићемо сваким елементом десно од њега, који пронађемо да је мањи. Променљива j ће у итерацији ,која следи, узимати све вредности индекса почев од i+1 па до n-1, што значи да се у једном пролазу кроз подниз почев од (i+1)-ог елемента тражи најмањи.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
classProgram
    {
static void Main(string[] args)
        {
//Navodimo direktno elemente niza a
int a[] = {2,5,6,8,4,9,3,7,11,10};
int i,j,pom;
//Uzimamo duzinu niza
int n = a.Length;
//Na primer postavicemo promenljivu i da bude na trecoj poziciji u nizu
            i = 2;

//Promenljivu j postavljamo od sledece pozicije u nizu
for(j = i+1;j < n;j++)

//Trazimo elemente a[j] koji su manji od fiksiranog elementa na trecoj poziciji
if(a[j] < a[i])
                {
//Svaki pronadjeni manji element zamenjujemo elementom na i-toj(trecoj) poziciji
pom = a[i];
a[i] = a[j];
a[j] = pom;
                }
            }
        }
    }
}
```

Понављањем претходног поступка за i=0,1,2,.....n-1 елементи низа ће се поређати у **растући поредак**.

На сличан начин можемо добити и низ сортиран у **опадајућем** редоследу, с тим што тада вршимо размену сваког i-тог елемента са сваким следећим елементом који је **већи**.

Постоје различите врсте алгоритма за сортирање, који се битно разликују по принципу рада и брзини извршавања.Ми ћемо се упознати са најједноставнијим **selectionsort** алгоритмом.

Баш као и у помоћном примеру, низ се сортира растуће тако што се најпре тражи **најмањи елемент**, који се доводи на прво место.Затим се налази најмањи од преосталих n-1 елемената и он се доводи на друго место, па најмањи у поднизу од n-2 елемента доводимо на треће место и тако закључно са налажењем мањег од последња два елемента и његовим довођењем на претпоследње место.На последњем месту ће остати елемент који није мањи ни од једног у низу (највећи елемент).

У наставку је приказан алгоритам сортирања у растућем поретку, у оквиру функције selection_sort која нема повратну вредност.Јасно је да та функција врши само пермутације и упоређивања елемената низа, па самим тим не враћа конкретан резултат добијен неким операцијама.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
classProgram
    {
static void selection_sort (int []a ,int n)
        {
int i,j,pom;
//Cuva indeks tekuceg minimalnog elementa
int min;

/*Spoljasnja for petlja koja se izvrsava prolaskom kroz niz po promenljivoj i,
                  tek kada se zavrsi unutrasnja for petlja*/
for(i = 0;i < n-1;i++)
            {
//Promenljivoj min dodelimo indeks elementa na i-toj poziciji
min = i;
/*Unutrasnja for petlja krece od narednog elementa, (i+1)-og. Kako se j povecava
sve se vise smanjuje podniz elemenata koje uporedjujemo sa i-tim elementom*/
for(j = i+1; j < n ;j++)
//Uporedjujemo tekuci element a[j] sa fiksiranim elementom na poziciji min
if(a[j] < a[min])
/*Kada pronadjemo novi manji element a[j], njegov indeks j dodeljujemo
promenljivoj min koja uvek cuva indeks tekuceg najmanjeg elementa*/
min = j;

/*Vrsimo zamenu mesta novog minimalnog elementa na poziciji min i elementa na
prethodnoj min poziciji, ako min vise nije i*/
if(min != i){
      pom = a[i];
      a[i] = a[min];
      a[min] = pom;
                }
            }
        }

static void Main(string []args)
        {
      //Rezervisemo proizvoljno 50 elemenata u memoriji
      int []a = newint[50];
      int i,n;

      Console.WriteLine("Unosimo broj elemenata niza:");
            n = int.Parse(Console.ReadLine());

      Console.WriteLine("Unosimo elemente niza:");

      for(i = 0;i < n;i++)

      a[i] = int.Parse(Console.ReadLine());

      //Pozivamo funkciju selection_sort koja ce nam sortirati uneti niz
            selection_sort(a , n);

//Ispisujemo elemente u novom rastucem poretku

      for(i = 0;i < n;i++)
      Console.WriteLine(a[i]+ ",");
        }
    }
}
```

Слично се врши и сортирање низа у опадајући поредак,са једном битном разликом. Којом?

Када низ сортирамо у опадајући поредак тражи се највећи елемент у сваком поднизу до последњег елемента, тако да ми уствари вршимо избор **узастопних максимума**, док смо при уређењу чланова низа растуће бирали узастопне минимуме. Покушајте сада сами да направите **selection_sort** алгоритам који сортира низ опадајуће.

Само пратите претходни поступак. 

Увек можете да преконтролишете свој код, ако ваш програм не ради након покретања и не можете да нађете грешку.

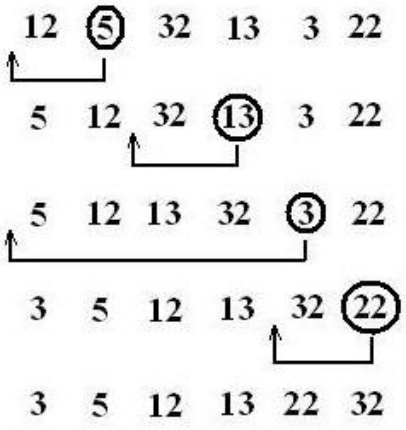
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
classProgram
    {
static void selection_sort_opada(int a[], int n)
        {
int i,j,pom;
//Cuva indeks tekuceg maksimalnog elementa
int max;
/*U slucaju opadajuceg poretka, vrsimo zamene kada pronadjemo
veci element od fiksiranog na i-toj poziciji*/
for(i = 0;i < n-1;i++)
            {
max = i;

for(j = i+1;j < n;j++)
if(a[j] > a[max])
max = j;

if(max != i)
                {
pom = a[i];
a[i] = a[max];
a[max] = pom;
                }
            }
        }

static void Main(string []args)
        {
int a[] = new int[50];
int n;

//Unosimo proizvoljan broj elemenata npr.10
Console.WriteLine("Unosimo broj elemenata niza:");
Console.ReadLine(n);

Console.WriteLine("Unosimo elemente niza:");

for(i = 0;i < n;i++)

a[i] = int.Parse(Console.ReadLine());

            selection_sort_opada(a , n);

//Ispisujemo elemente niza u novom poretku
for(i = 0;i < n;i++)
Console.WriteLine(a[i] + ",");
        }
```

```
    }
}
```

Овде ћемо се осврнути на још један алгоритам за сортирање низова, **insertion sort**.Овај алгоритам се користи код низова које треба "мало" сортирати и низова малих димензија.

Insertion sort алгоритам за сортирање у растућем поретку ради тако што пролази редом кроз сваки елемент у низу и проверава да ли је елемент пре њега (лево од њега) већи, и ако је већи врши се замена елемената и тако све док постоји елемент на левој страни који је већи од њега.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace insert_sort
{
classProgram
    {
static void insertion_sort(int[] a, int n)
        {
int i, pom, t;
for (i = 0; i < n; i++)
            {
      /*t postavimo na tekuci indeks i zatim pod uslovom da je t>0 uporedjujemo element
                  sa prethodno poredjanim elementima*/
                t = i;
while (t > 0 && a[t] < a[t - 1])
                {
//Zamenjujemo vrednosti elemenata na prethodnoj poziciji sa manjim na poziciji t
pom = a[t];
a[t] = a[t - 1];
a[t - 1] = pom;
/*Posto smo izvrsili zamenu elemenata t se smanjuje, pomera se na poziciju ulevo
i tako se while petljom  uporedjuju svaka dva prethodna elementa u podnizu*/
t--;
                }
/*Kada uslovi while petlje vise nisu zadovoljeni, izvrsava se spoljasnja for petlja
i t dobija novu vrednost*/
            }
        }

      static void Main(string[] args)
        {
int []a = new int[50];
int i, n;


Console.WriteLine("Unosimo broj elemenata niza:");
            n = int.Parse(Console.ReadLine());

Console.WriteLine("Unosimo elemente niza:");

for(i = 0;i < n;i++)
            {
a[i] = int.Parse(Console.ReadLine());
            }
            insertion_sort(a, n);

//Ispisujemo elemente u rastucem poretku

for(i = 0;i < n;i++)
Console.Write(a[i]+ " ");
        }

    }
}
```



Пример 1. Написати програм који сортира унети низ произвољне дужине и рачуна медијан, тј средњи члан тог низа.Ако је број чланова низа непаран, медијан је средњи члан, а ако је број чланова низа паран медијан је средња вредност елемената низа са индексиман/2-1 и n/2 где је nброј чланова низа.

Решење:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace medijan
{
classProgram
    {
//Funkcija koja sortira elemente niza u rastucem poretku
static void selection sort(int[] a, int n)
        {
int i, j, pom;
int min;
for (i = 0; i < n - 1; i++)
            {
min = i;

for (j = i + 1; j < n; j++)
if (a[j] < a[min])
min = j;

if(min != i)
                {
pom = a[i];
a[i] = a[min];
a[min] = pom;
                }
```

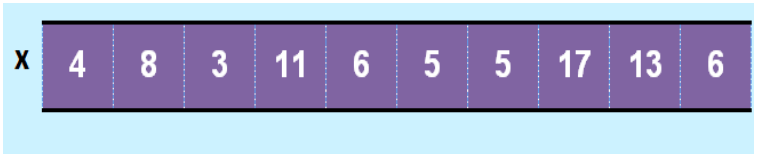
```
            }
        }

static void Main(string[] args)
        {
int[] a=new int[100];
int i, j, dim;

//Sa ulaza unosimo broj clanova niza a[] i tu vrednost dodeljujemo promenljivoj dim
Console.WriteLine("Unesite broj clanova niza");
dim = int.Parse(Console.ReadLine());

//Sa ulaza unosimo clanove niza a[]
Console.WriteLine("Unesite clanove niza");
for (i = 0; i < dim; i++)
            {
a[i] = int.Parse(Console.ReadLine());
            }

//Pozivamo funkciju selection_sort
            selection_sort(a,dim);
for (i = 0; i < dim; i++)
            {
//Ispisujemo sortirani niz
Console.WriteLine(" " +a[i]);
            }

//Postavimo medijan na pocetnu vrednost
int med = 0;

//Racunamo medijan niza a prema gore navedenom pravilu

//Ako je broj clanova niza neparan, tj.dim nije deljiv sa 2
if (dim % 2 != 0)
            {
//Medijan je srednji clan, ciji je indeks (dim-1)/2
med = a[(dim - 1) / 2];
            }
//Ako je broj clanova niza paran, tj.ako je dim deljivo sa 2
else if (dim % 2 == 0)
            {
//Medijan racunamo kao aritmeticku sredinu dva elementa na pomenutim pozicijama
med = (a[(dim / 2)-1]+a[(dim/2)])/2;
            }
//Ispisujemo rezultat
Console.WriteLine("Medijan je:", med);
```

```
        }
    }
}
```

Појам дводимензионог низа

До сада смо имали прилике да се упознамо са једнодимензионим низовима чији су елементи скаларне величине (цели бројеви, реални бројеви, знакови,..). Овакви низови се могу шематски приказати као хоризонтална или вертикална листа података, што можемо видети на следећим сликама.

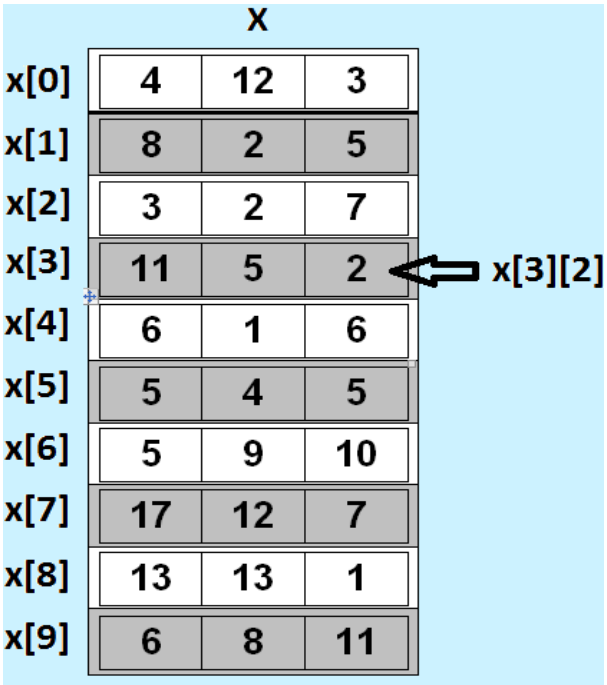
x
4
8
3
11
6
5
5
17
13
6



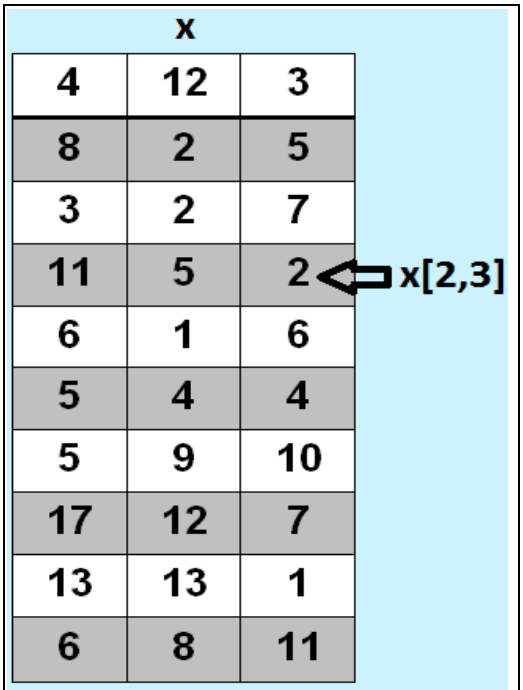
Слика 10.14. Врста

Слика 10.13. Колона

Међутим, у великом броју реалних проблема је погодније податке представити у табеларној форми.

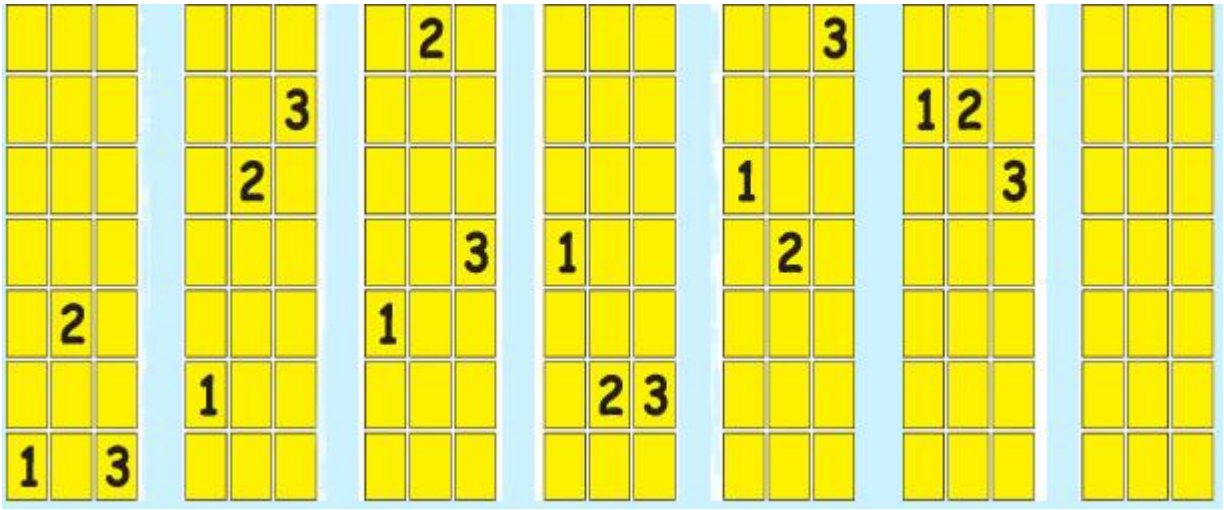


Слика 10.15. Низ низова



Слика 10.16. Матрица

Табела на слици 10.15, висине 10 и ширине 3 садржи 10 врста и 3 колоне. Произвољна табела висине *m* и ширине *n*, може се представити као низ дужине *m* чији су елементи **низови** дужине *n*, у ствари имамо *m* врста и *n* колона. Зато дводимензионе низове најчешће називамо **матрице** (слика 8.4).



Погледајмо како би изгледале декларације и дефиниције неких матрица. Могуће је приликом дефинисања придружити и почетне вредности.

```
//Deklaracije proizvoljnih matrica
int [ , ] aMatrica;
float [ , ] bMatrica;

//Matrica od 10 celobrojnih nizova duzine 3
int [ , ] aMatrica = new int[10,3];
//Matrica od 8 nizova tipa float duzine 2
float [ , ] bMatrica = new float[8,2];

//Matrica dimenzija 3 x 4 (3 reda, 4 kolone)
int[ , ] intMatrica =
{
    //Vrednosti nultog reda
    {1, 2, 3, 4},
    //Vrednosti prvog reda
    {5, 6, 7, 8},
    //Vrednosti drugog reda
    {9, 10, 11, 12},
};
```

Сада се можда питате како би могли да приступимо одређеном елементу матрице?! **Појединачној врсти** се може приступити коришћењем записа ***aMatrica[i]***, где је ***i*** редни број врсте.

Елементу у *i*-тој врсти и *j*-тој колони се може приступити коришћењем записа ***aMatrica[i ,j]***.

У случају да наша матрица има велики број елемената најбоље је унети вредности са тастатуре у једној ***for*** петљи.

```
//Definisemo matricu intMatrica dimenzija 3×4
int [ , ] intMatrica = new int[3, 4];
int i,j;

Console.WriteLine("Unesi clanove matrice:");

/*Spoljasnja for petlja prolazi po vrstama. Svaka vrsta po i predstavlja jedan niz,
ciji se elementi unose po j.*/
for (i = 0; i < 3; i++)
{
    /*Unutrasnja for petlja prolazi po kolonama. Kada je jedna vrsta fiksirana, npr.
za i=0,
    unose se elementi po promenljivoj j, npr. intMatrica[0,j]*/
    for (j = 0; j < 4; j++)
    {
        /*Sledeca naredba ispisuje naziv svakog elementa pojedinacno,pre ispisa
vrednosti
        intMatrica[i,j]*/
        Console.Write("clan[",i,j,"] = ");
        intMatrica[ i, j] = int.Parse(Console.ReadLine());
    }
}

//Ispisujemo unete clanove po vrstama i kolonama
for(i = 0; i < 3; i++)
{
    for (j = 0; j < 4 ; j++)
    {
        //Navodnici ostavljaju prazan prostor posle svakog unetog elementa
        Console.Write(" " + intMatrica[i ,j]);
    }
    //Kada se ispise jedna vrsta, prelazi se u sledecu i tako se formira matricni
oblik
    Console.WriteLine();
}
```

Стартовањем овог програма и уносом вредности чланова добија се конзолни екран приказан на следећој слици:

```
Unesi clanove matrice:
clan[0,0] = 1
clan[0,1] = 2
clan[0,2] = 3
clan[0,3] = 4
clan[1,0] = 5
clan[1,1] = 6
clan[1,2] = 7
clan[1,3] = 8
clan[2,0] = 9
clan[2,1] = 10
clan[2,2] = 11
clan[2,3] = 12
 1 2 3 4
 5 6 7 8
 9 10 11 12
Press any key to continue . . .
```

Слика 10.18. Конзола

Напомена: Метод *GetLength* има увек повратну вредност типа *int* и враћа број елемената некога низа. У нашој итерацији учествује при провери услова да се бројачи увек крећу по индексима редова и врста матрице.

Пример 1. Написати програм који учитава матрицу целих бројева *a* димензије $m \times n$ и рачуна суму свих елемената матрице.

Решење:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace suma_elemenata_matrice
{
    class Program
    {
        static void Main(string[] args)
        {
            //Rezervisemo proizvoljan broj vrsta i kolona u memoriji
            int[,] a = new int[ 30,40 ];
            int i, j;
            //m i n kao dimenzije matrice
            int m, n;
            int s = 0;

            //Unosimo dimenzije matrice
```

```
            Console.WriteLine("Unesite broj vrsta:");
            m = int.Parse(Console.ReadLine());

            Console.WriteLine("Unesite broj kolona:");
            n = int.Parse(Console.ReadLine());

            //Unosimo elemente matrice po vrstama
            Console.WriteLine("Unesi clanove matrice:");

            for (i = 0; i < m; i++)
            {
                for (j = 0; j < n; j++)
                {
                    Console.Write("clan[",i,j,"] = ");
                    a [i, j] = int.Parse(Console.ReadLine());
                }
            }
            //Racunamo sumu svih elemenata matrice
            for (i = 0; i < m; i++)
            {
                for (j = 0; j < n; j++)
                {
                    //Iteracijom po promenljivoj j kroz svaku i-tu vrstu dodajemo
elemente u zbir s
                    s = s + a[i, j];
                }
            }

            Console.WriteLine("Suma elemenata matrice je:" + s);
        }
    }
}
```

Пример 2. **Написати програм који учитава квадратну матрицу *a* целих бројева димензије *n* × *n* и израчунава највећи и најмањи елемент у матрици, као и збир елемената на главној дијагонали.**

Напомена: Главну дијагоналу матрице чине елементи на дијагонали, која спаја горњи леви и доњи десни угао матрице.

Решење:

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace max_min_matrice
{
    class  Program
    {
        //Funkcija koja racuna min i max matrice
         static void min_max(int[,] a,  int n)
        {

            int i, j;
            //Postavljamo promenljive max i min na pocetne vrednosti
            //Element a[0,0] se nalazi u gornjem levom uglu date matrice a
            int max = a[0, 0];
            int min = a[0, 0];
            //Spoljasnjom for petljom prolazimo kroz matricu a po vrstama
            for (i = 0; i < n; i++)
            {
                //Unutrasnjom for petljom prolazimo kroz matricu a po kolonama
                for (j = 0; j < n; j++)
                {
                    //Proveravamo da li je a[i,j] manji od tekuceg minimuma
                    if (a[i, j] < min)
                    {
                        //Ako je manji, onda njegovu vrednost dodeljujemo promenljivoj
min
                        min = a[i, j];
                    }

                    /*U slucaju da je a[i,j] veci od tekuceg maksimuma, njegovu
vrednost dodeljujemo
                    promenljivoj max*/
                    if (a[i, j] > max)
                    {
                        max = a[i, j];
                    }
                }
            }
            Console.WriteLine("Najveci  element matrice je:", max,"a najmanji je:",
min);
        }

        //Funkcija dijagonala izracunava zbir elemenata matrice na dijagonali
        static void dijagonala(int[,] a, int n)
        {
            int i, suma = 0;
```

```
            //Kako je i=j dovoljna nam je jedna for petlja po vrstama i kolonama
matrice a
            for (i = 0; i < n; i++)
            {
                //Clanovi na dijagonali su oblika a[i,i]
                suma += a[i, i];
            }
            Console.WriteLine("Suma  elemenata na glavnoj dijagonali je {0}", suma);
        }

        static void Main(string[] args)
        {
            int[,] aMatrica = new int[10, 10];
            int i, j, max, min, dim;

            Console.WriteLine("Unesite dimenziju matrice");
            dim = int.Parse(Console.ReadLine());

            Console.WriteLine("Unesite elemente matrice");
            for (i = 0; i < dim; i++)
            {
                for (j = 0; j < dim; j++)
                {
                    Console.Write("clan[",i,j,"] = ");
                    aMatrica[i, j] = int.Parse(Console.ReadLine());
                }
            }

            //Standardno ispisujemo elemente unete matrice iteracijom po vrstama i
kolonama
            for (i = 0; i < dim; i++)
            {
                for (j = 0; j < dim; j++)
                {
                    Console.Write(aMatrica[i, j]);
                }
                Console.Write('\n');
            }
            //Pozivamo funckiju koja racuna min i max matrice aMatrica
            min_max(aMatrica, dim);
            //Pozivamo funckiju koja racuna zbir elemenata na dijagonali matrice
aMatrica
            //Ispis rezultata se vrsi pri pozivu funkcija
            dijagonala(aMatrica, dim);

        }
```

```
    }
}
```

Пример 3. **Написати програм којим се за задату квадратну матрицу *A* димензије *n* × *n*, израчунава квадрат те матрице (*A* × *A*).**

Решење:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace kvadrat_matrice
{
    class  Program
    {
        //Funkcija kvadriraj izracunava vrednost matrice  A*A
        static void kvadriraj(int[,] A, int n)
        {
            /*Definisemo kvadratnu matricu B dimenzija n, koja ce predstavljati
            rezultat kvadriranja matrice A*/
            int[,] B = new int[n, n];
            int i, j, k;

            //Matricu B dobijamo tako sto mnozimo odgovarajuce vrste i kolone matrice
A
            for (i = 0; i < n; i++)
                for (j = 0; j < n; j++)
                {
                    /*Pocetnu vrednost elementa matrice B na preseku i-te vrste i j-te
kolone
                    postavljamo na 0, kako tekuci element dobijamo sabiranjem n
proizvoda
                    elemenata matrice A*/
                    B[i, j] = 0;
                    /*Pomocna for petlja po promenljivoj k omogucava da se istovremeno
pomnoze
                    elementi u i-toj vrsti i j-toj koloni matrice A*/
                    for (k = 0; k < n; k++)
                    {
                        /*Proizvoljni element B[i,j] dobijamo tako sto medjusobno
pomnozimo elemente
                        i-te vrste i j-te kolone matrice A*/
                        B[i, j] += A[i, k] * A[k, j];
                    }
                }
            //Ispisujemo elemente rezultujuce matrice B
```

```
            Console.WriteLine("Matrica B = A*A:");
            for (i = 0; i < n; i++)
            {
                for (j = 0; j < n; j++)
                {
                    Console.Write(" " + B[i, j]);
                }
                Console.WriteLine();
            }
        }

        static void Main(string[] args)
        {
            //Definisemo matricu A proizvoljno izabrane dimenzije
            int[,] A = new int [20,20];
            int dim, i, j;

            Console.WriteLine("Unesite dimenziju kvadratne matrice");
            dim = int.Parse(Console.ReadLine());

            //Unosimo elemente matrice A
            for (i = 0; i < dim; i++)
            {
                for (j = 0; j < dim; j++)
                {
                    Console.Write("clan[",i,j,"] = ");
                    A[i, j] = int.Parse(Console.ReadLine());
                }
            }
            //Pozivamo funkciju kvadriraj  koja ce ispisati dobijenu kvadriranu
matricu A*A
            kvadriraj(A,dim);
        }
    }
}
```

Пример 4. **Учитати природан број *n* и квадратну матрицу *A* реда *n*. Формирати транспоновану матрицу дате матрице *A*.**

Напомена: Транспонована матрица се добија када одговарајуће врсте и колоне замене места.

Решење:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace transponovana_matrica
{
    class  Program
    {
        //Funkcija transponuj formira transponovanu matricu date matrice A dimenzije n
         static void transponuj(int[,] A, int n)
        {
            int i, j, pom;

            /*Prolazimo for petljom kroz elemente matrice A i vrsimo zamenu
odgovarajucih vrsta
            i kolona, npr. prva kolona postaje prva vrsta itd.*/
            for (i = 0; i < n; i++)
            {
                for (j = 0; j < i; j++)
                {
                    //Vrsimo zamenu elemenata na poziciji A[i,j] elementima na
poziciji A[j,i]
                    pom = A[i, j];
                    A[i,j] = A[j, i];
                    A[j, i] = pom;
                }
            }

            //Ispisujemo matricu u transponovanom obliku
            Console.WriteLine("Transponovana matrica ");
            for (i = 0; i < n; i++)
            {
                for (j = 0; j < n; j++)
                {
                    Console.Write( " "  + a[i, j]);
                }
                Console.WriteLine();
            }
        }

        //Glavna funkcija Main u kojoj unosimo proizvoljnu matricu poziva funkciju
transponuj
        static void Main(string[] args)
        {
            int[,] aMatrica = new int[15,15];
```

```
            int i, j, dim;

            Console.WriteLine("Unesite dimenziju kvadratne matrice");
            dim = int.Parse(Console.ReadLine());

            Console.WriteLine("Unesi elemente matrice:");

            for (i = 0; i < dim; i++)
            {
                for (j = 0; j < dim; j++)
                {
                    //clan[i,j]= vrednost aMatrica[i,j]
                    Console.Write("clan[",i,j,"] = ");
                    aMatrica[i, j] = int.Parse(Console.ReadLine());
                }
            }
            //Pozivamo funkciju koja vrsi transponovanje unete matrice aMatrica
            transponuj(aMatrica,dim);

        }
    }
}
```

Пример 5. **Учитати матрицу димензија *n* × *m*. Сортирати елементе по врстама и колонама унете матрице, а затим исписати тако формирану матрицу.**

Решење:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace sortiranje elemenata
{
    class  Program
    {
        static void sortiranje_kolona(int[,] matrica,  int n,  int m)
        {
            int i, j;
            int pom, t;
            for (j = 0; j < m; j++)
            {
                //Sortiramo kolone uz pomoc insertion sort algoritma koji nam je
poznat od ranije
                for (i = 0; i < n; i++)
                {
                    t = i;
                    while (t > 0 && matrica[t, j] < matrica[t-1, j])
                    {
```

```
                        pom = matrica[t, j];
                        matrica[t, j] = matrica[t - 1, j];
                        matrica[t - 1, j] = pom;
                        t--;
                    }
                }
            }

            //Ispisujemo matricu ciji su elementi u svakoj koloni sortirani
            Console.WriteLine("Matrica cije su kolone sortirane");
            for (i = 0; i < n; i++)
            {
                for (j = 0; j < m; j++)
                {
                    Console.Write(matrica[i, j]);
                }
                Console.WriteLine('\n');
            }
        }

        static void sortiranje_vrsta(int[,] matrica,int n,int m)
        {
            int i, j, t;
            int pom;
            for (i = 0; i < n; i++)
            {
                for (j = 0; j < m; j++)
                {
                    t = j;
                    while(t > 0 && matrica[i, t]< matrica[i, t-1])
                    {
                        //Clan na poziciji [i,t] menja mesto sa clanom na poziciji
[i,t-1]
                        pom = matrica[i, t];
                        matrica[i,t] = matrica[i, t-1];
                        matrica[i, t-1] = pom;
                        t--;
                    }
                }
            }

            //Ispisujemo matricu kod koje su elementi u svakoj vrsti sortirani
            Console.WriteLine("Matrica cije su vrste sortirane");
            for (i = 0; i < n; i++)
            {
```

```
                for (j = 0; j < m; j++)
                {
                    Console.Write(matrica[i, j]);
                }
                Console.WriteLine('\n');
            }
        }
        static void Main(string[] args)
        {
            int[,] A = new int[50, 50];
            int i, j, m, n;
            Console.WriteLine("Unesite broj vrsta matrice");
            n = int.Parse(Console.ReadLine());
            Console.WriteLine("Unesite broj kolona matrice");
            m = int.Parse(Console.ReadLine());
            for (i = 0; i < n; i++)
            {
                for (j = 0; j < m; j++)
                {
                    Console.Write("clan[{0},{1}] = ", i, j);
                    A[i, j] = int.Parse(Console.ReadLine());
                }
            }
            sortiranje_vrsta(A, n, m);
            sortiranje_kolona(A, n, m);
        }
    }
}
```

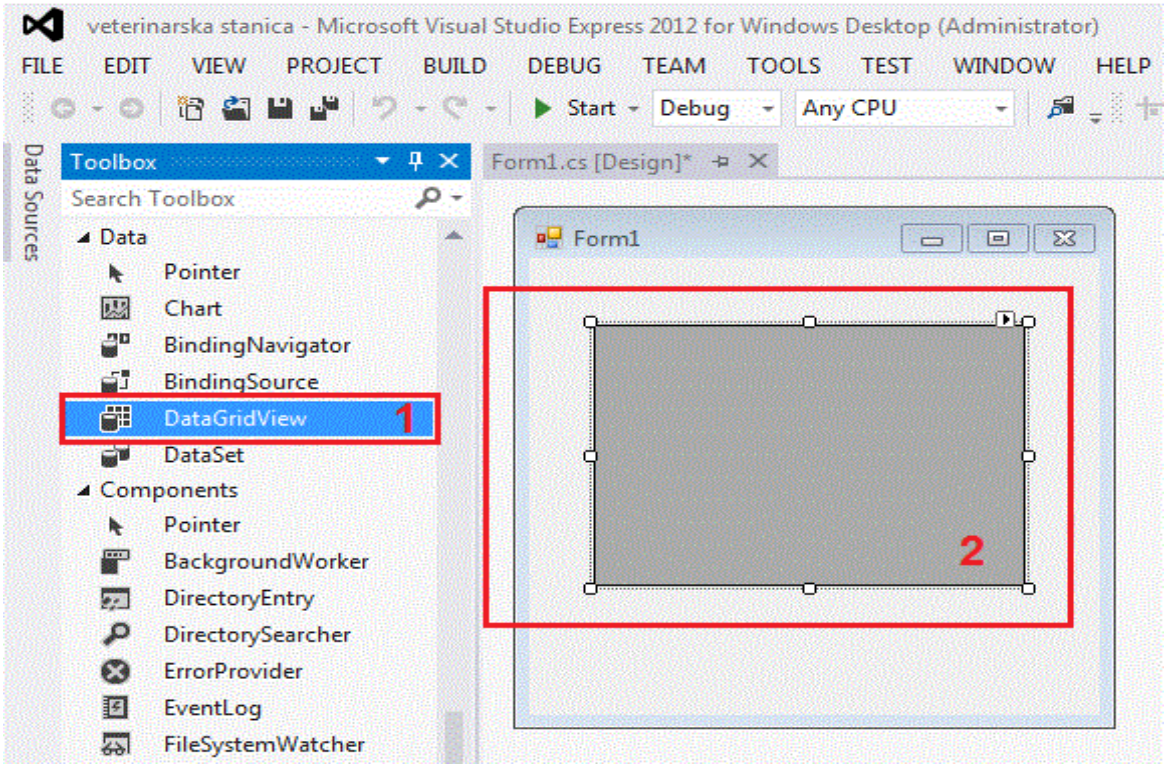
Визуелна компонента за табеларни приказ текста

Ако имате кућног љубимца сигурно сте га некада одвели код ветеринара. Шта мислите како у ветеринарској станици успевају да воде евиденцију о свим посетама ваших чупавих пријатеља? Једноставно у Visual C# апликацији !

Ако желимо да неки скуп елемената прикажемо у посебно дизајнираној табели можемо то урадити помоћу специјалне компоненте ***DataGridView***, коју ћете пронаћи у *C#* палети са алаткама – *ToolBox* у секцији *Data*. Реч је о *GUI* елементу *C#* програмског окружења који омогућава сваком кориснику да при раду са низовима и матрицама визуелно лепше и прецизније представи како улазне податке тако и резултат. Наравно, сваки нови унос се лако подешава.

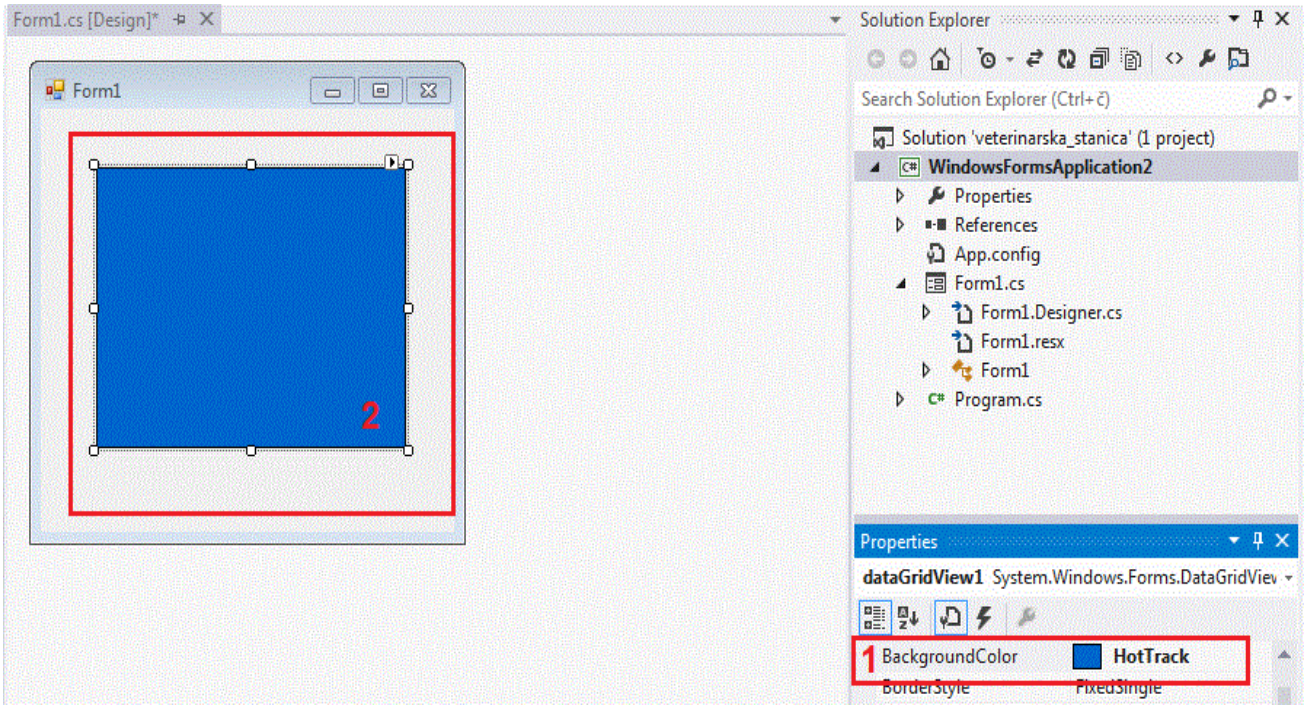
Сада ћемо се упознати са основним корацима формирања једног табеларног приказа у виду апликације на примеру наше ветеринарске станице.

Отворимо нови пројекат, где нам се појави *visual* окружење, након што смо задали име пројекта. Двоструким кликом или превлачењем додамо *DataGridView* компоненту на *Form-y*.



Слика 10.20. Радно окружење

У *Properties Windows* можемо подесити неке параметре као што је боја позадине, ивица табеле, итд. Ми ћемо само да променимо BackGroundColor, нпр. у плаву чисто због лепшег изгледа.

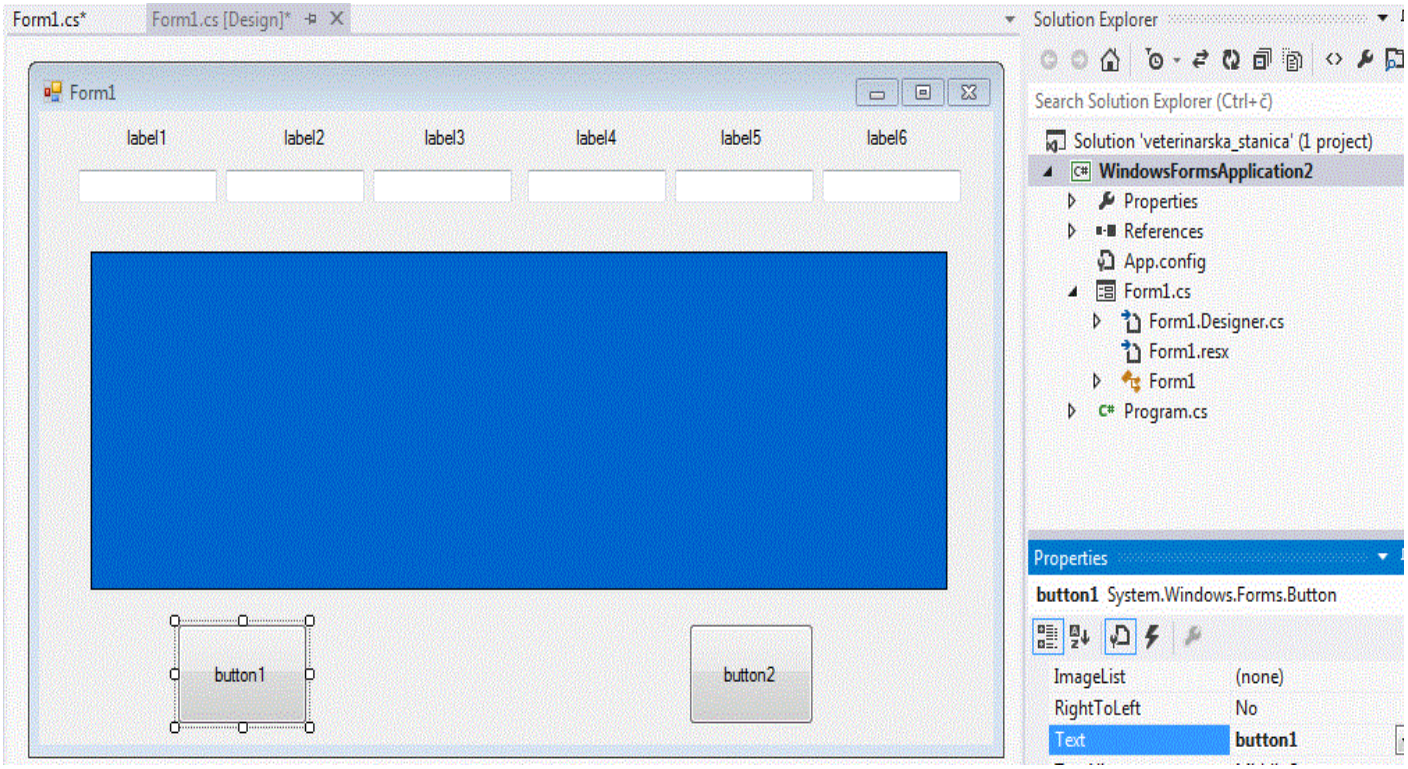


Слика 10.21. Подешавања форме

Пошто корисник (у нашем примеру ветеринар) пре прегледа сваког пацијента уноси нове податке, биће нам потребна нека текстуална поља за њихово брже и ефикасније уношење, као и лабеле које ће представљати тачно име податка у одређеном пољу. Додаћемо и два дугмета:

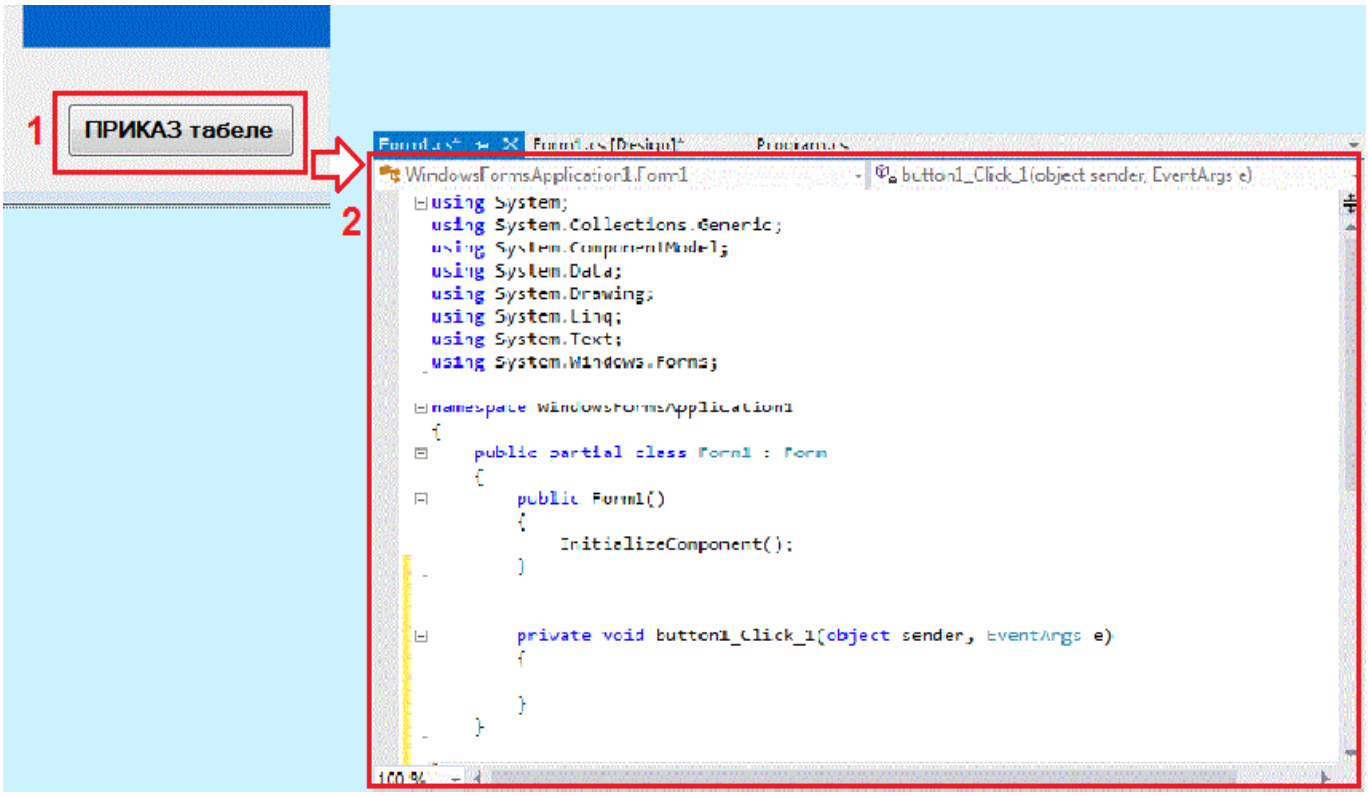
за *приказ* табеле – када корисник кликне појави се табела са до тада унетим параметрима

- сачувај – пошто смо унели податке о љубимцу у текстуална поља, кликом на ово дугме они се сачувају у одговарајућим пољима табеле



Слика 10.22. Визуелни изглед форме

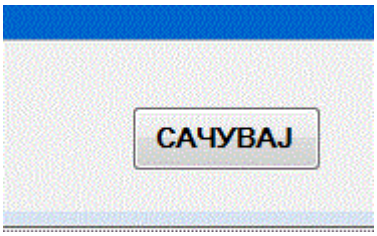
Сада нам још остаје да испрограмирамо акције за нашу дугмад! Двоструким кликом на свако од дугмади отвара се прозор Form1.cs, где пишемо код, који ће покретањем програма (F5) да извршава оно што желимо.



Слика 10.23. Акција првог дугмета

Овако би изгледао код који извршава функцију првог дугмета:

```
private void button1_Click(object sender, EventArgs e)
{
  private void button1_Click(object sender, EventArgs e)
{
    /*Pravimo tabelu koja ce sadrzati sledece podatke:
    ime,vrstu,sifru,tezinu,boju i datum pregleda ljubimca*/

    //Zadajemo 6 kolona
    dataGridView1.ColumnCount = 6;


    //Sada zadajemo nazive kolona pojedinacno, vodeci racuna da indeks kolone uvek
krece od 0

    dataGridView1.Columns[0].Name = "Име љубимца";
    dataGridView1.Columns[1].Name = "Врста љубимца";
    dataGridView1.Columns[2].Name = "Старост";
    dataGridView1.Columns[3].Name = "Тежина";
    dataGridView1.Columns[4].Name = "Боја";
    dataGridView1.Columns[5].Name = "Датум прегледа";
}
```



Код за друго дугме чије извршавање чува сваки нови унос корисника:

Слика 10.24. Дугме за чување уноса

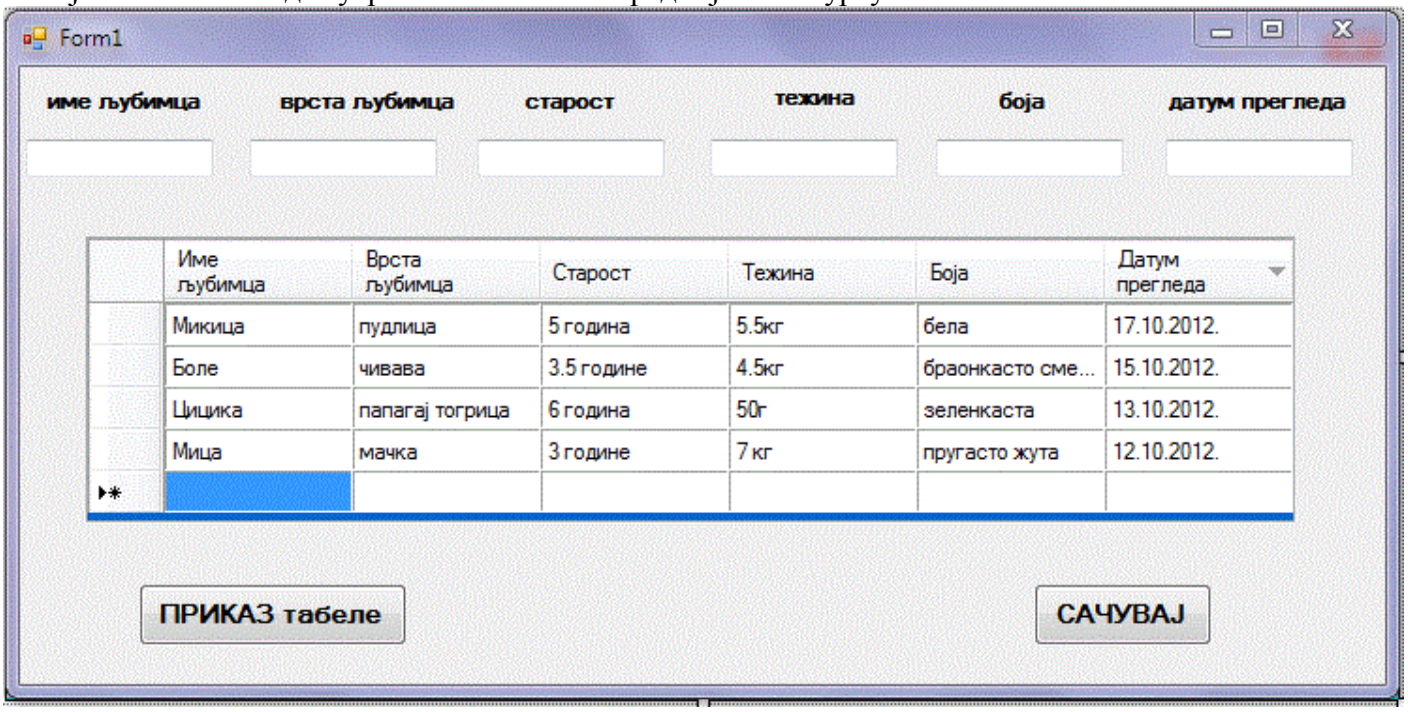
```
private void button2_Click(object sender, EventArgs e)
{
    string ime, vrsta, starost, tezina, boja, datum;

    //Postavljamo string promenljive na vrednosti unete preko textBoxova

    ime = textBox1.Text;
    vrsta = textBox2.Text;
    starost = textBox3.Text;
    tezina = textBox4.Text;
    boja = textBox5.Text;
    datum = textBox6.Text;

    //Sve podatke koji ce biti u jednoj vrsti stavljamo u niz stringova
```

```
    String[] red = new String[6] { ime, vrsta, starost, tezina, boja, datum };

    //Sada ove podatke unosimo u tabelu po vrstama
    dataGridView1.Rows.Add(red);

    //Zatim brisemo iz textBoxova prethodno unete podatke

    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
    textBox5.Text = "";
    textBox6.Text = "";
}
```

Напомена: Зашто желимо да нам се сваки пут обришу претходно унете информације о пацијенту? Да би се поља за унос аутоматски ослободила и омогућила кориснику да несметано уноси нове податке, без потребе да најпре обрише старе податке из *textBoxova*.

И ево табеле, на основу које у ветеринарској станици увек располажу тачним подацима о нашим љубимцима Овакав распред подсећа на једну базу података,

са којом ћете се можда сусрести на неком напреднијем C# курсу.



Слика 10.25. Табеларни приказ ветеринарске станице