

SVEUČILIŠTE/UNIVERZITET „VITEZ“

FAKULTET INFORMACIONIH TEHNOLOGIJA

STUDIJSKOG ciklusa; GODINA studija: I; III

SMJER: INFORMACIJSKE TEHNOLOGIJE



Zulka Musić

20 ZADATAKA U C# SA OBRAZLOŽENJEM

SEMINARSKI RAD

Travnik, 16.12.2024. godine

SVEUČILIŠTE/UNIVERZITET „VITEZ“

FAKULTET INFORMACIONIH TEHNOLOGIJA

STUDIJSKOG ciklusa; GODINA studija: I; III

SMJER: INFORMACIJSKE TEHNOLOGIJE



20 ZADATAKA U C# SA OBRAZLOŽENJEM

SEMINARSKI RAD

IZJAVA: Ja **Zulka Musić**, studentica Sveučilišta/Univerziteta „VITEZ“, Indeks broj: **390-24/RIIT** odgovorno i uz moralnu i akademsku odgovornost izjavljujem da sam ovaj rad izradila potpuno samostalno uz korištenje citirane literature i pomoć predmetnog profesora.

STUDENT: Zulka Musić

PREDMET: Napredne .NET tehnologije

MENTOR: doc.dr. Mahir Zajmović

ASISTENT: dr.sc. Bakir Čičak

SADRŽAJ

1.	UVOD.....	1
1.1.	PROBLEM, PREDMET I OBJEKT ISTRAŽIVANJA.....	1
1.2.	SVRHA I CILJEVI ISTRAŽIVANJA.....	1
1.3.	RADNA HIPOTEZA I POMOĆNE HIPOTEZE	1
1.4.	NAUČNE METODE	2
1.5.	STRUKTURA RADA	2
2.	ZADATAK 1	3
2.1.	IZRADA ZADATKA 1	4
2.2.	OBRAZLOŽENJE ZADATKA 1.....	6
3.	ZADATAK 2	9
3.1.	IZRADA ZADATKA 2	9
3.2.	OBRAZLOŽENJE ZADATKA 2.....	10
4.	ZADATAK 3	11
4.1.	IZRADA ZADATKA 3	11
4.2.	OBRAZLOŽENJE ZADATKA 3.....	12
5.	ZADATAK 4.....	13
5.1.	IZRADA ZADATKA 4	13
5.2.	OBRAZLOŽENJE ZADATKA 4.....	14
6.	ZADATAK 5	15
6.1.	IZRADA ZADATKA 5	15
6.2.	OBRAZLOŽENJE ZADATKA 5.....	16
7.	ZADATAK 6	17
7.1.	IZRADA ZADATKA 6	17
7.2.	OBRAZLOŽENJE ZADATKA 6.....	18

8.	ZADATAK 7	19
8.1.	IZRADA ZADATKA 7	19
8.2.	OBRAZLOŽENJE ZADATKA 7.....	20
9.	ZADATAK 8	21
9.1.	IZRADA ZADATKA 8	21
9.2.	OBRAZLOŽENJE ZADATKA 8.....	22
10.	ZADATAK 9	23
10.1.	IZRADA ZADATKA 9	23
10.2.	OBRAZLOŽENJE ZADATKA 9.....	24
11.	ZADATAK 10	25
11.1.	IZRADA ZADATKA 10	25
11.2.	OBRAZLOŽENJE ZADATKA 10.....	26
12.	ZADATAK 11	27
12.1.	IZRADA ZADATKA 11	27
12.2.	OBRAZLOŽENJE ZADATKA 11.....	28
13.	ZADATAK 12	29
13.1.	IZRADA ZADATKA 12	29
13.2.	OBRAZLOŽENJE ZADATKA 12.....	30
14.	ZADATAK 13	31
14.1.	IZRADA ZADATKA 13	31
14.2.	OBRAZLOŽENJE ZADATKA 13.....	32
15.	ZADATAK 14	33
15.1.	IZRADA ZADATKA 14	33
15.2.	OBRAZLOŽENJE ZADATKA 14.....	34
16.	ZADATAK 15	35

16.1.	IZRADA ZADATKA 15	35
16.2.	OBRAZLOŽENJE ZADATKA 15.....	36
17.	ZADATAK 16	37
17.1.	IZRADA ZADATKA 16	37
17.2.	OBRAZLOŽENJE ZADATKA 16.....	37
18.	ZADATAK 17	38
18.1.	IZRADA ZADATKA 17	38
18.2.	OBRAZLOŽENJE ZADATKA 17.....	38
19.	ZADATAK 18	39
19.1.	IZRADA ZADATKA 18	39
19.2.	OBRAZLOŽENJE ZADATKA 18.....	39
20.	ZADATAK 19	40
20.1.	IZRADA ZADATKA 19	40
20.2.	OBRAZLOŽENJE ZADATKA 19.....	41
21.	ZADATAK 20	42
21.1.	IZRADA ZADATKA 20	42
21.2.	OBRAZLOŽENJE ZADATKA 20.....	43
22.	ZAKLJUČAK	44

1. UVOD

1.1. PROBLEM, PREDMET I OBJEKT ISTRAŽIVANJA

Ovaj seminarski rad bavi se proučavanjem osnovnih i naprednih funkcionalnosti programskog jezika C#. Problem istraživanja je kako kroz praktične primjere zadataka ilustrirati ključne koncepte i metode rada u C#. Predmet istraživanja su zadaci i rješenja, dok je objekt istraživanja primjena C# u razvoju softverskih rješenja. Svi kodovi urađenih zadataka će se nalaziti u ovom radu kao i na github-u.

Link: <https://github.com/Zulka12/CSharp-Zadaci-Seminarski>

1.2. SVRHA I CILJEVI ISTRAŽIVANJA

Svrha rada je unaprijeđenje razumijevanja i vještine rada sa C# kroz implementaciju 20 praktičnih zadataka. Glavni cilj je da se kroz rješavanje zadataka. Dodatni cilj je povezivanje teorijskih znanja sa praksom.

1.3. RADNA HIPOTEZA I POMOĆNE HIPOTEZE

Radna hipoteza je da se osnovni i napredni koncepti C# mogu uspješno savladati kroz rješavanje praktičnih zadataka. Pomoćne hipoteze uključuju tvrdnje da se kroz ovakav pristup može poboljšati logičko razmišljanje, razumjevanje algoritama i optimizacija koda.

1.4. NAUČNE METODE

U radu su korištene metode analize, sinteze i komparacije. Analizom se identifikuju problemi i rješavaju zadaci, sintezom se stvara kod na osnovu zadatka, dok komparacija omogućava poređenje različitih rješenja u cilju pronalaska optimalnog.

1.5. STRUKTURA RADA

Struktura seminarskog rada je usklađena sa Uputstvom za pisanje seminarskog rada na prvom ciklusu studija kao i temi seminarskog rada. On sadrži dvadeset i dva poglavlja.

- Prvo poglavlje, Uvod, sadrži pet pod poglavlja:
 - Problem, predmet i objekt isprašivanja,
 - Svrha i ciljevi istraživanja,
 - Radna hipoteza i pomoćne hipoteze,
 - Naučne metode,
 - Struktura rada.
- Drugo poglavlje kao i ostala poglavlja u radu sadrži dva pod poglavlja, svako poglavlje nosi isto ime samo je drugi broj zadatka:
 - Izrada zadatka,
 - Objašnjenje zadatka.
- Dvadeset i drugo poglavlje, Zaključak, daje odgovore na postavljene hipoteze.

2. ZADATAK 1

Napišite konzolnu aplikaciju u programskom jeziku C#, koja simulira upravljanje podacima o studentima na fakultetu. Aplikacija treba da omogući korisniku sljedeće funkcionalnosti:

Dodavanje studenata:

- Korisnik unosi ime, prezime i broj indeksa studenta.
- Nakon unosa, novi student se dodaje u listu studenata.

Prikaz svih studenata:

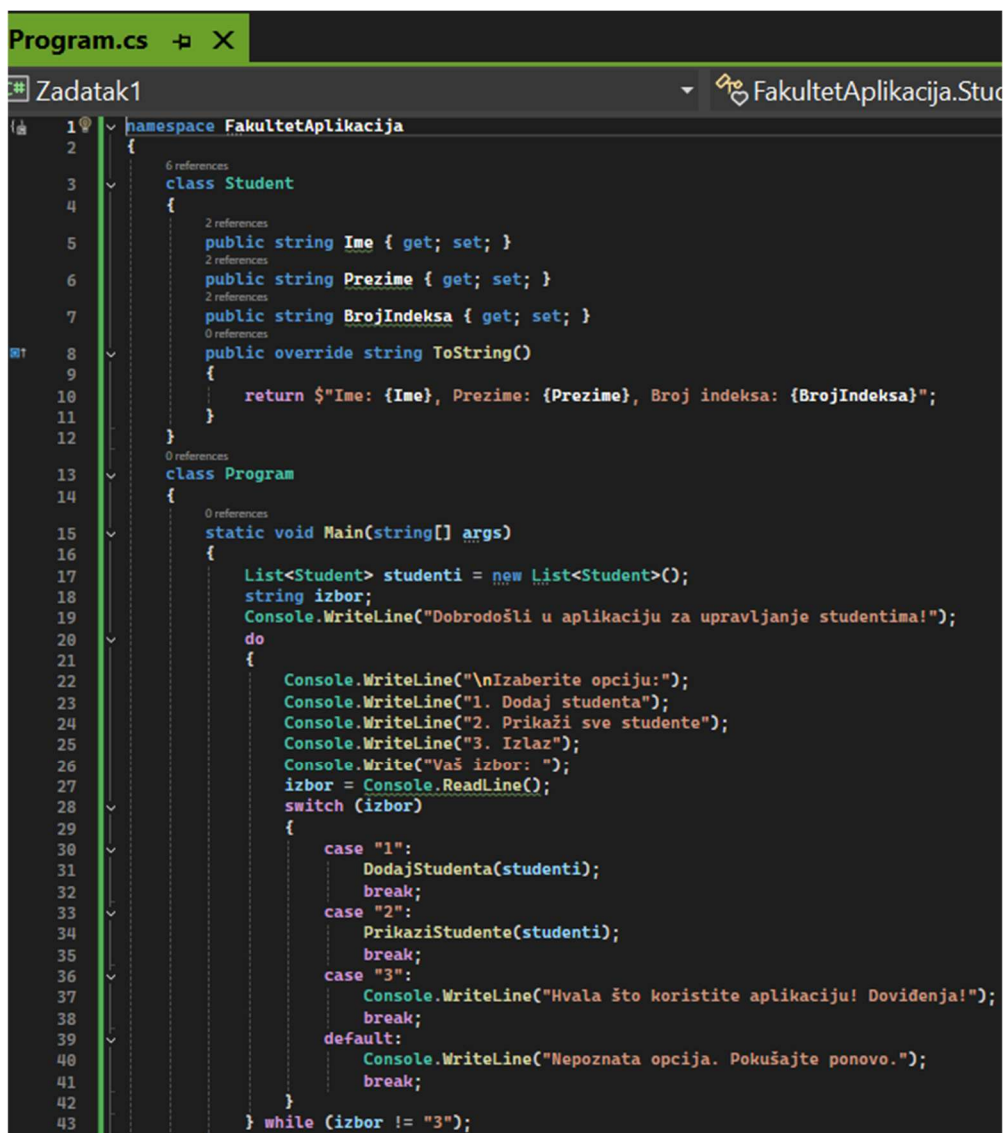
- Prikazuje listu svih unijetih studenata sa njihovim podacima (ime, prezime i broj indeksa).
- Ako lista studenata nije popunjena, aplikacija treba da obavijesti korisnika da nema unijetih studenata.

Izlaz iz aplikacije:

- Korisnik može da prekine rad aplikacije.

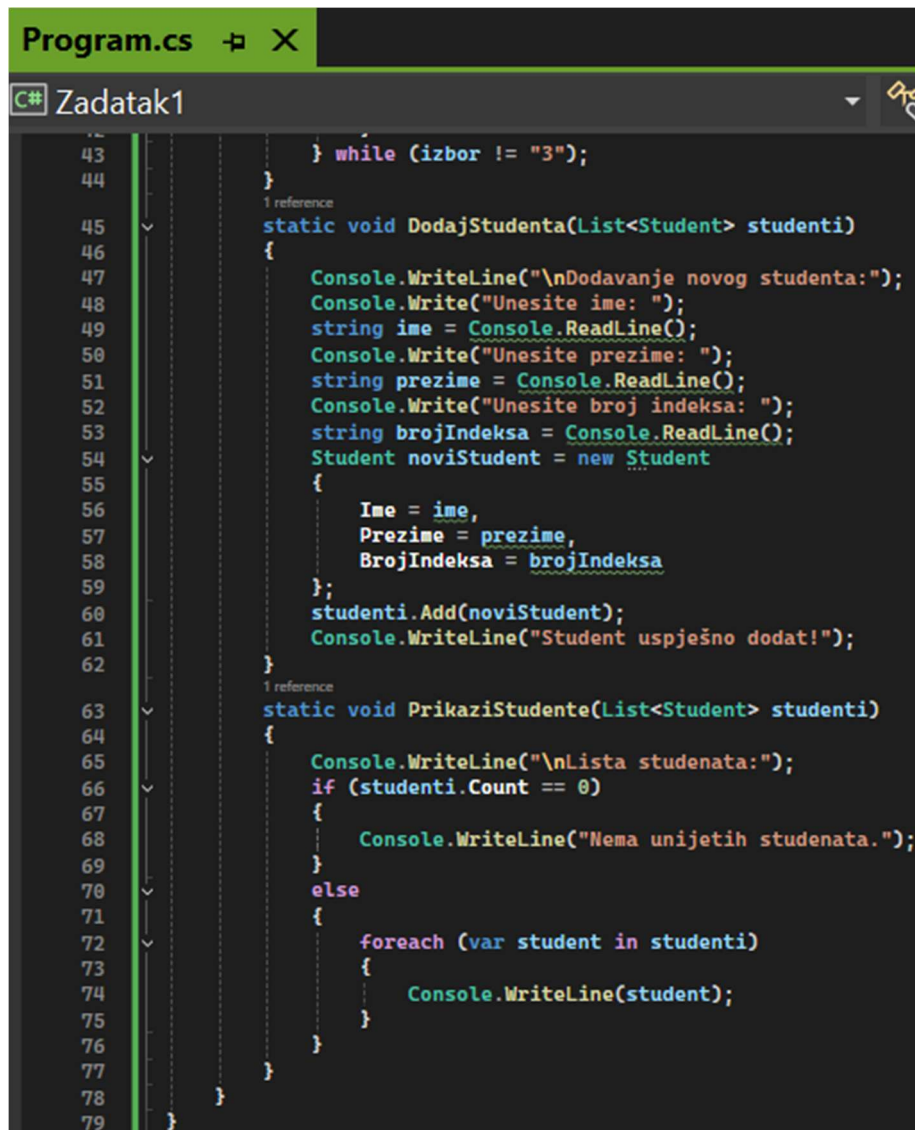
2.1. IZRADA ZADATKA 1

Slika 1. Zadatak 1



```
Program.cs X
Zadatak1
namespace FakultetAplikacija
{
    class Student
    {
        public string Ime { get; set; }
        public string Prezime { get; set; }
        public string BrojIndeksa { get; set; }
        public override string ToString()
        {
            return $"Ime: {Ime}, Prezime: {Prezime}, Broj indeksa: {BrojIndeksa}";
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            List<Student> studenti = new List<Student>();
            string izbor;
            Console.WriteLine("Dobrodošli u aplikaciju za upravljanje studentima!");
            do
            {
                Console.WriteLine("\nIzaberite opciju:");
                Console.WriteLine("1. Dodaj studenta");
                Console.WriteLine("2. Prikaži sve studente");
                Console.WriteLine("3. Izlaz");
                Console.Write("Vaš izbor: ");
                izbor = Console.ReadLine();
                switch (izbor)
                {
                    case "1":
                        DodajStudenta(studenti);
                        break;
                    case "2":
                        PrikažiStudente(studenti);
                        break;
                    case "3":
                        Console.WriteLine("Hvala što koristite aplikaciju! Doviđenja!");
                        break;
                    default:
                        Console.WriteLine("Nepoznata opcija. Pokušajte ponovo.");
                        break;
                }
            } while (izbor != "3");
        }
    }
}
```

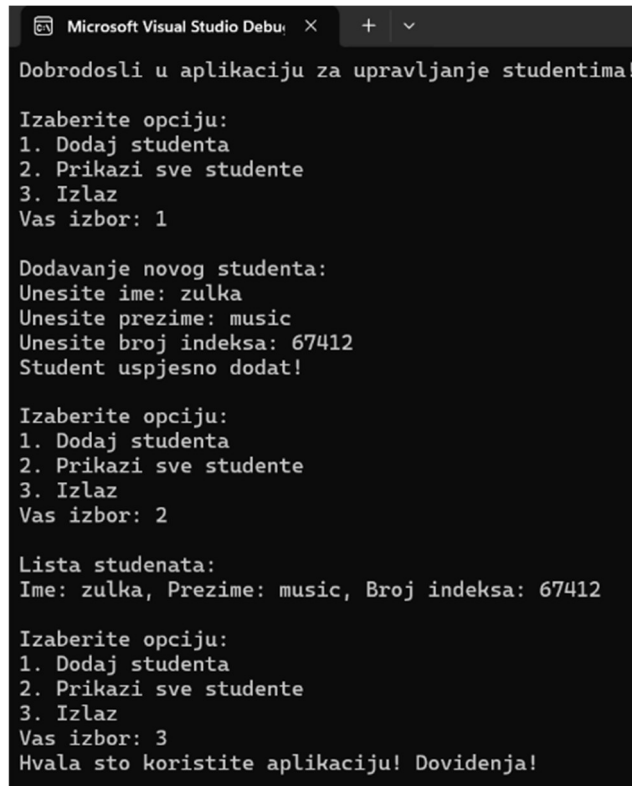
Slika 2. Zadatak 1



```
Program.cs  X
C# Zadatak1

43     } while (izbor != "3");
44     }
45     static void DodajStudenta(List<Student> studenti)
46     {
47         Console.WriteLine("\nDodavanje novog studenta:");
48         Console.Write("Unesite ime: ");
49         string ime = Console.ReadLine();
50         Console.Write("Unesite prezime: ");
51         string prezime = Console.ReadLine();
52         Console.Write("Unesite broj indeksa: ");
53         string brojIndeksa = Console.ReadLine();
54         Student noviStudent = new Student
55         {
56             Ime = ime,
57             Prezime = prezime,
58             BrojIndeksa = brojIndeksa
59         };
60         studenti.Add(noviStudent);
61         Console.WriteLine("Student uspješno dodat!");
62     }
63     static void PrikaziStudente(List<Student> studenti)
64     {
65         Console.WriteLine("\nLista studenata:");
66         if (studenti.Count == 0)
67         {
68             Console.WriteLine("Nema unijetih studenata.");
69         }
70         else
71         {
72             foreach (var student in studenti)
73             {
74                 Console.WriteLine(student);
75             }
76         }
77     }
78 }
79 }
```

Slika 3. Output zadatka 1



```
Microsoft Visual Studio Debu: X + v
Dobrodosli u aplikaciju za upravljanje studentima!
Izaberite opciju:
1. Dodaj studenta
2. Prikazi sve studente
3. Izlaz
Vas izbor: 1

Dodavanje novog studenta:
Unesite ime: zulka
Unesite prezime: music
Unesite broj indeksa: 67412
Student uspjesno dodat!

Izaberite opciju:
1. Dodaj studenta
2. Prikazi sve studente
3. Izlaz
Vas izbor: 2

Lista studenata:
Ime: zulka, Prezime: music, Broj indeksa: 67412

Izaberite opciju:
1. Dodaj studenta
2. Prikazi sve studente
3. Izlaz
Vas izbor: 3
Hvala sto koristite aplikaciju! Dovidjenja!
```

2.2. OBRAZLOŽENJE ZADATKA 1

Na prvom zadatku ću objasniti sve dijelove koda, a za sve ponavljane dijelove u kasnijim zadacima objašnjenje neću pisati da ne bi rad bio predugačak. Iako napišem u Visual Studio-u `using System` i `using System.Collections.Generic` prilikom pokretanja zadatka Visual Studio taj dio minimizira, da se ne vidi u kodu, zbog toga u kodu gore nema tog dijela, ali ću ih u nastavku objasniti.

- `using System;` - Omogućava korištenje osnovnih klasa i metoda iz .NET biblioteke, kao što su Console za unos i ispis podataka na konzolu.
- `using System.Collections.Generic;` - Omogućava korištenje generičkih kolekcija, uključujući `List<T>` koja se koristi za skladištenje podataka o studentima.
- `namespace FakultetAplikacija` – Definiše prostor imena koji grupiše sve klase u ovoj aplikaciji. Koristi se za izbjegavanje konflikata imena sa klasama iz drugih biblioteka.

- `class Student`: Definiše klasu `Student` koja predstavlja podatke jednog studenta.
- `public string Ime { get; set; }` – Automatska svojstva za čuvanje imena studenta. Omogućavaju čitanje i pisanje vrijednosti pomoću ključeva `get` i `set`. Slično za `Prezime` i `BrojIndeksa`.
- `public override string ToString()` – Nadjačava (`override`) podrazumijevanu implementaciju metode `ToString`, koja vraća `string` reprezentaciju objekta.
- `return $"Ime: {Ime}, Prezime: {Prezime}, Broj indeksa: {BrojIndeksa}";` - Formatira `string` sa podacima studenta.
- `class Program` – Glavna klasa aplikacije koja sadrži metodu `Main`.
- `static void Main(string[] args)` – Početna tačka aplikacije. Kada se pokrene program, izvršavanje počinje od `Main`-a.
- `List<Student>` - Lista koja čuva objekte klase `Student`.
- `new List<Student>()` – Kreira praznu listu koja će tokom rada programa skladištiti informacije o studentima.
- `string izbor;` - Promenljiva za skladištenje izbora korisnika iz menija.
- `Console.WriteLine` – Ispisuje tekst na konzolu.
- `do { } while` – Petlja koja se ponavlja sve dok korisnik ne izabere opciju za izlaz (3).
- `Console.WriteLine` – Ispisuje meni sa opcijama.
- `Console.Write` – Ispisuje prompt za unos izbora korisnika.
- `izbor = Console.ReadLine();` - Čita unos korisnika sa konzole.
- `switch (izbor)` – Grananje koje odlučuje koju akciju će aplikacija izvršiti na osnovu korisnikovog izbora.
- `case "1"` – Poziva metodu `DodajStudenta` za dodavanje novog studenta.
- `case "2"` – Poziva metodu `PrikaziStudente` za prikaz svih studenata.
- `case "3"` – Prekida petlju i završava aplikaciju.
- `Default` – Obrada nepoznatog unosa, ispisuje poruku o grešci.
- `List<Student> studenti` – Lista koja se prosleđuje metodi `i` u kojoj se čuvaju podaci o studentima.
- `Console.Write` – Ispisuje prompt za unos imena.


- `Console.ReadLine` – Čita uneseno ime sa konzole.
- `new Student` – Kreira novi objekat klase `Student` i dodjeljuje vrijednosti za ime, prezime i broj indeksa.
- `studenti.Add` – Dodaje novog studenta u listu `studenti`.
- `Console.WriteLine`: Obaveštava korisnika o uspešnom dodavanju.
- `static void PrikaziStudente(List<Student> studenti)` – Definiše metodu za prikaz svih unijetih studenata.
- `if (studenti.Count == 0)` – Provjerava da li je lista prazna.
- `foreach (var student in studenti)` – Iterira kroz sve studente u listi i ispisuje njihove podatke koristeći nadjačanu metodu `ToString`.

3. ZADATAK 2

Napisati program u programskom jeziku C# koji demonstrira koncept nasljeđivanja korištenjem dvije klase: Zivotinja (bazna klasa) i Pas (nasljedna klasa). Program treba omogućiti da u baznoj klasi Zivotinja se definira Ime za pohranu imena životinje i metodu KreciSe() koja se ispisuje u formatu [Ime] se kreće. U nasljednoj klasi Pas se treba kreirati metoda Laj() i ispisuje se [Ime] laje: 'Av, av!'.

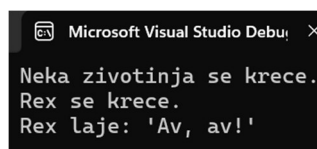
3.1. IZRADA ZADATKA 2

Slika 4. Zadatak 2



```
1 namespace Nasljedjivanje
2 {
3     class Zivotinja
4     {
5         public string Ime { get; set; }
6         public void KreciSe()
7         {
8             Console.WriteLine($"{Ime} se kreće.");
9         }
10    }
11    class Pas : Zivotinja
12    {
13        public void Laj()
14        {
15            Console.WriteLine($"{Ime} laje: 'Av, av!'");
16        }
17    }
18    class Program
19    {
20        static void Main(string[] args)
21        {
22            Zivotinja zivotinja = new Zivotinja();
23            zivotinja.Ime = "Neka Zivotinja";
24            zivotinja.KreciSe();
25            Pas pas = new Pas();
26            pas.Ime = "Rex";
27            pas.KreciSe();
28            pas.Laj();
29            Console.ReadLine();
30        }
31    }
32 }
```

Slika 5. Output zadatka 2



```
Microsoft Visual Studio Debu: X
Neka zivotinja se krece.
Rex se krece.
Rex laje: 'Av, av!'
```

3.2. OBRAZLOŽENJE ZADATKA 2

Ovo je primjer jednostavne implementacije nasljeđivanja.

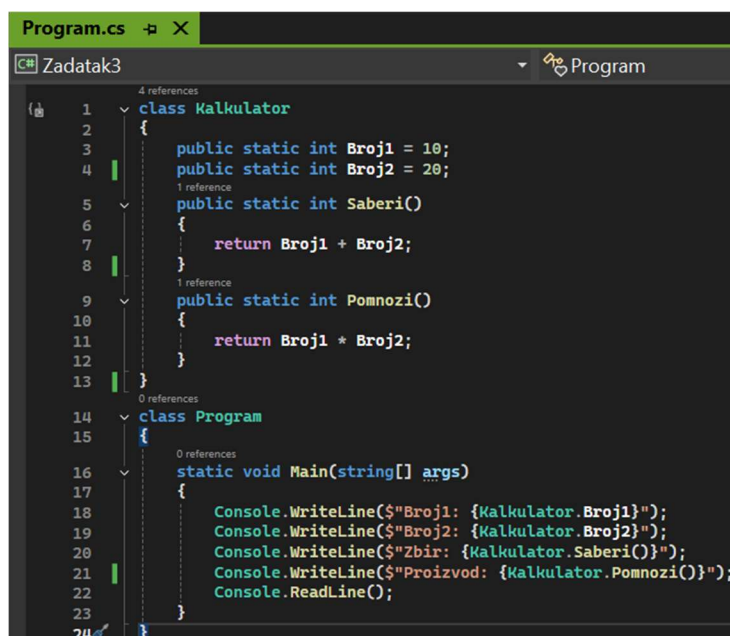
Zivotinja je bazna klasa koja predstavlja generičku životinju. Ime je javno i omogućava pohranu imena životinje, automatski je implementirano pomoću get i set. Metoda KreciSe ispisuje na konzolu poruku da se životinja kreće, koristeći vrijednost osobine Ime. Klasa Pas nasljeđuje klasu Zivotinja koristeći dvotačku (:). Svi javni i zaštićeni članovi klase Zivotinja automatski postaju dostupni u klasi Pas. Metoda Laj je specifična za klasu Pas i omogućava psu da „laje“. Pristupa naslijeđenoj osobini Ime kako bi prilagodila poruku. U Main funkciji se demonstrira kako koristiti klase Zivotinja i Pas. Kreira se instanca bazne klase (Zivotinja) i nasljedne klase (pas). Postavlja se osobina Ime i poziva metoda KreciSe u oba slučaja, te metoda specifična za klasu Pas se također poziva.

4. ZADATAK 3

Napisati program u C# jeziku koji koristi statička polja i statičke metode za implementaciju jednostavnog kalkulatora. Program treba da ima klasu Kalkulator sa statičkim poljima i metodama koje izvedu matematičke operacije sabiranje i množenje.

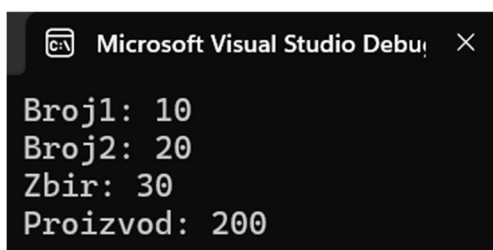
4.1. IZRADA ZADATKA 3

Slika 6. Zadatak 3



```
Program.cs X
Zadatak3 Program
1 class Kalkulator
2 {
3     public static int Broj1 = 10;
4     public static int Broj2 = 20;
5     public static int Saberi()
6     {
7         return Broj1 + Broj2;
8     }
9     public static int Pomnozi()
10    {
11        return Broj1 * Broj2;
12    }
13 }
14 class Program
15 {
16     static void Main(string[] args)
17     {
18         Console.WriteLine($"Broj1: {Kalkulator.Broj1}");
19         Console.WriteLine($"Broj2: {Kalkulator.Broj2}");
20         Console.WriteLine($"Zbir: {Kalkulator.Saberi()}");
21         Console.WriteLine($"Proizvod: {Kalkulator.Pomnozi()}");
22         Console.ReadLine();
23     }
24 }
```

Slika 7. Output zadatka 3



```
Microsoft Visual Studio Debug Console X
Broj1: 10
Broj2: 20
Zbir: 30
Proizvod: 200
```


4.2. OBRAZLOŽENJE ZADATKA 3

Statička polja (Broj1 i Broj2) pripadaju klasi, a ne instancama te klase. To znači da im se može pristupiti bez kreiranja objekta klase, koristeći sintaksu `Kalkulator.Broj1`. Statičke metode (`Saberi()` i `Pomnozi()`) mogu direktno pristupiti statičkim poljima klase. One omogućuju izvođenje operacija koristeći podatke koji su zajednički za cijelu klasu. Budući da su polja i metode statički, nema potrebe za instanciranjem klase `Kalkulator`. Direktno se koristi ime klase za pristup članovima (npr. `Kalkulator.Saberi()`).

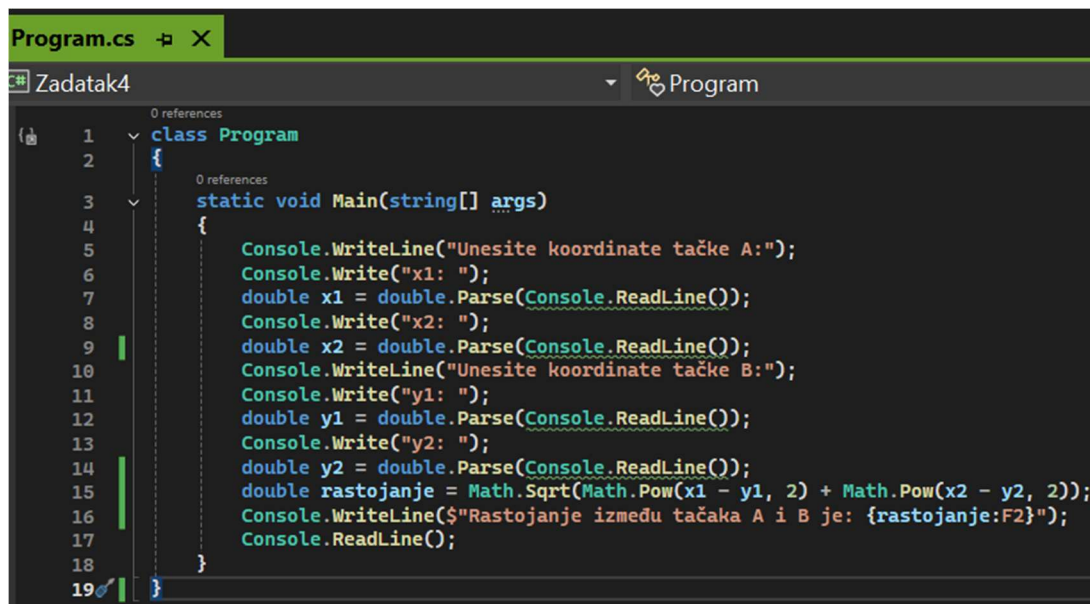
5. ZADATAK 4

Napisati c# program koji računa rastojanje dvije tačke u ravni ako tačka A ima koordinate (x1, x2), a tačka B ima koordinate (y1, y2), onda je rastojanje tačaka A i B dato sa

$$d(A, B) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

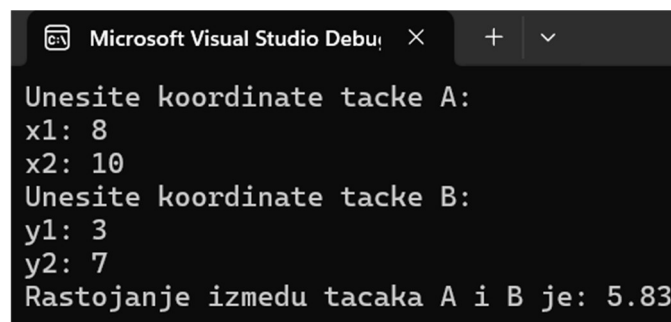
5.1. IZRADA ZADATKA 4

Slika 8. Zadatak 4



```
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.WriteLine("Unesite koordinate tačke A:");
6         Console.Write("x1: ");
7         double x1 = double.Parse(Console.ReadLine());
8         Console.Write("x2: ");
9         double x2 = double.Parse(Console.ReadLine());
10        Console.WriteLine("Unesite koordinate tačke B:");
11        Console.Write("y1: ");
12        double y1 = double.Parse(Console.ReadLine());
13        Console.Write("y2: ");
14        double y2 = double.Parse(Console.ReadLine());
15        double rastojanje = Math.Sqrt(Math.Pow(x1 - y1, 2) + Math.Pow(x2 - y2, 2));
16        Console.WriteLine($"Rastojanje između tačaka A i B je: {rastojanje:F2}");
17        Console.ReadLine();
18    }
19 }
```

Slika 9. Output zadatka 4



```
Microsoft Visual Studio Debug Console
Unesite koordinate tacke A:
x1: 8
x2: 10
Unesite koordinate tacke B:
y1: 3
y2: 7
Rastojanje između tacaka A i B je: 5.83
```

5.2. OBRAZLOŽENJE ZADATKA 4

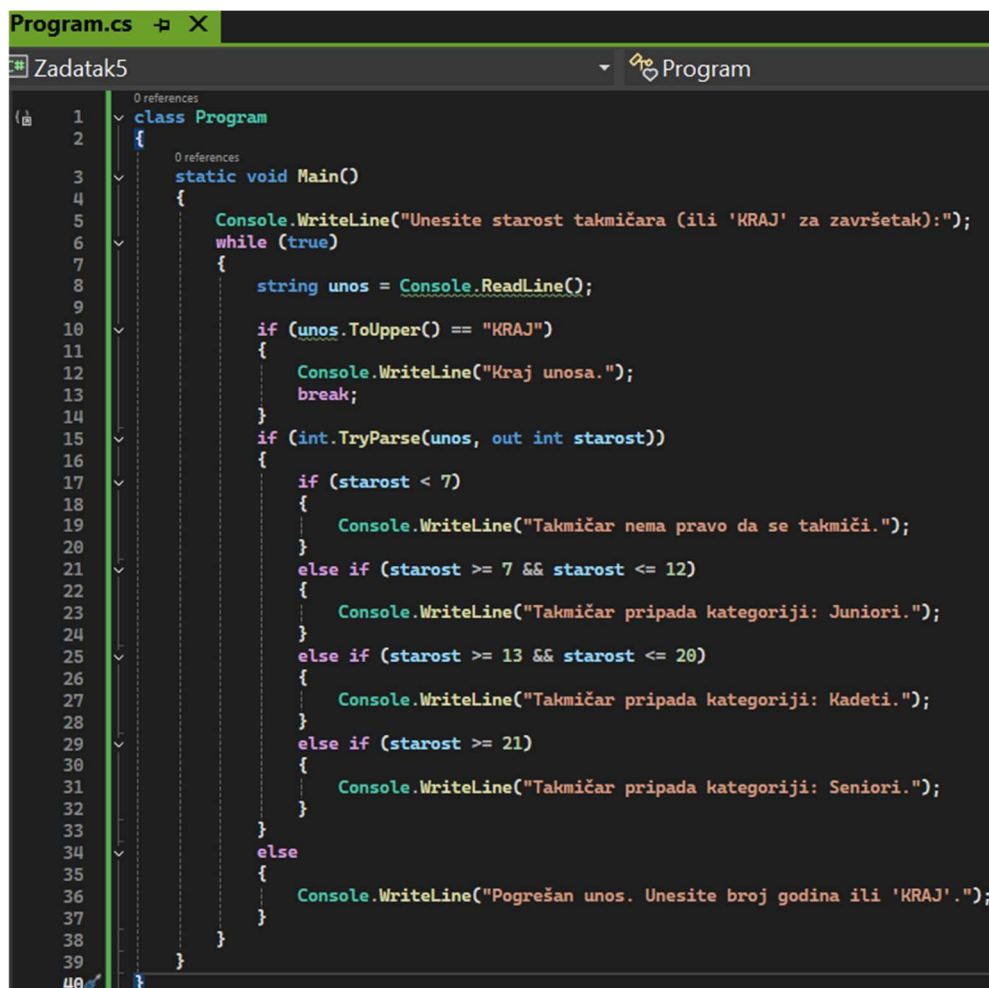
Ovaj program ima zadatak da izračuna rastojanje između dvije tačke u koordinatnom sistemu pomoću formule za udaljenost između tačaka. Korisnik unosi koordinate za tačku A i tačku B. Potom se izračunava udaljenost između tački A i B, gdje koristimo metodu `Math.Sqrt` za izračunavanje korijena, a metodu `Math.Pow` je za računanje kvadrata. Zatim nam izbacuje rezultat zaokružen na dvije decimale.

6. ZADATAK 5

Na jednom turniru karatisti se takmiče u tri kategorije: juniori (takmičari os 7 do 12 godina), kadeti (takmičari od 13 do 20 godina) i seniori (takmičari od 21 godinu na dalje). Napisati C# program koji od korisnika učitava cijele brojeve (koji predstavlja starost prijavljenih takmičara) sve dok korisnik ne unese riječ KRAJ, za svakog takmičara ispisuje kategoriju kojoj pripada. Ukoliko takmičar ima manje od 7 godina program treba da ispiše da takmičar nema pravno da se takmiči.

6.1. IZRADA ZADATKA 5

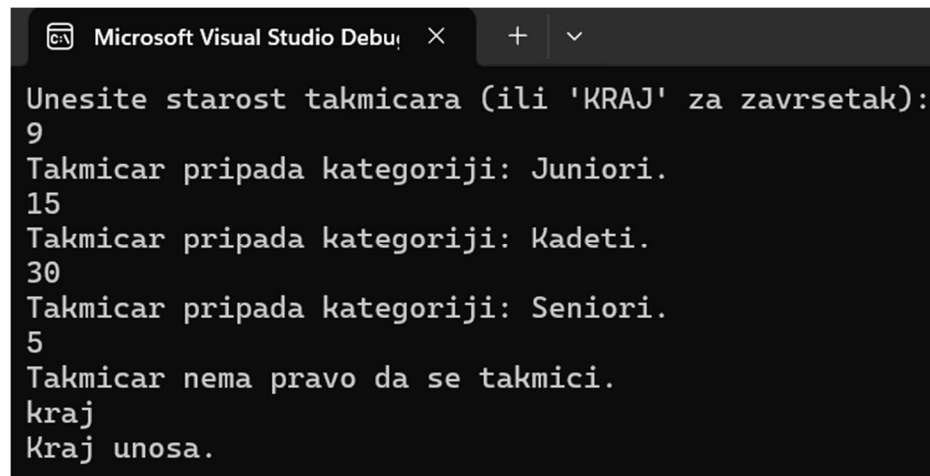
Slika 10. Zadatak 5



```
Program.cs
Zadatak5
class Program
{
    static void Main()
    {
        Console.WriteLine("Unesite starost takmičara (ili 'KRAJ' za završetak:");
        while (true)
        {
            string unos = Console.ReadLine();

            if (unos.ToUpper() == "KRAJ")
            {
                Console.WriteLine("Kraj unosa.");
                break;
            }
            if (int.TryParse(unos, out int starost))
            {
                if (starost < 7)
                {
                    Console.WriteLine("Takmičar nema pravo da se takmiči.");
                }
                else if (starost >= 7 && starost <= 12)
                {
                    Console.WriteLine("Takmičar pripada kategoriji: Juniori.");
                }
                else if (starost >= 13 && starost <= 20)
                {
                    Console.WriteLine("Takmičar pripada kategoriji: Kadeti.");
                }
                else if (starost >= 21)
                {
                    Console.WriteLine("Takmičar pripada kategoriji: Seniori.");
                }
            }
            else
            {
                Console.WriteLine("Pogrešan unos. Unesite broj godina ili 'KRAJ'.");
            }
        }
    }
}
```

Slika 11. Output zadatka 5



```
Microsoft Visual Studio Debu  ×  +  v
Unesite starost takmicara (ili 'KRAJ' za zavrsetak):
9
Takmicar pripada kategoriji: Juniori.
15
Takmicar pripada kategoriji: Kadeti.
30
Takmicar pripada kategoriji: Seniori.
5
Takmicar nema pravo da se takmici.
kraj
Kraj unosa.
```

6.2. OBRAZLOŽENJE ZADATKA 5

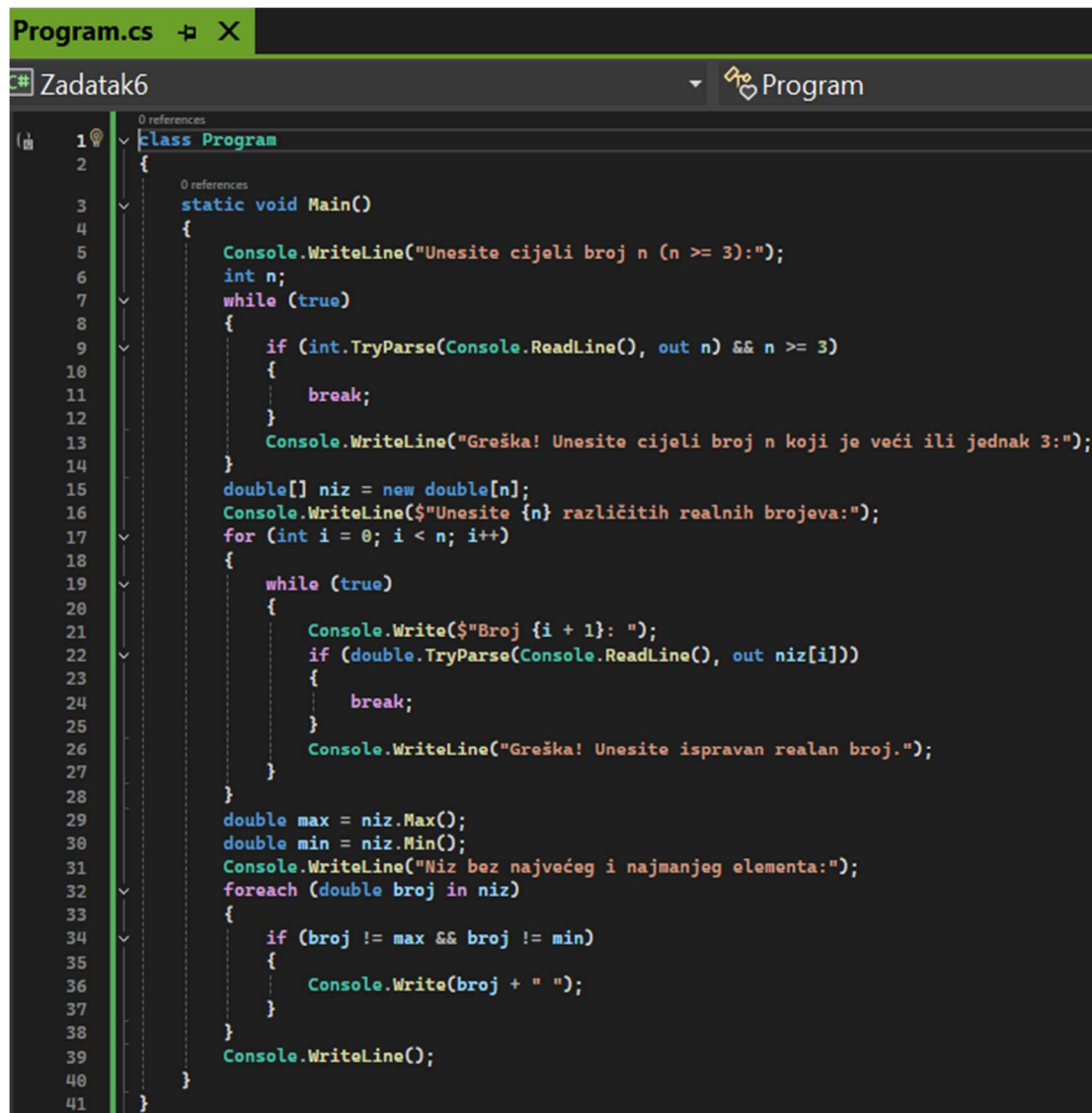
Prvo napišemo šta će nam program izbaciti kad pokrenemo isti, a to je da unesemo starost takmičara. Program će se ponavljati sve dok ne upišemo KRAJ, ToUpper() je funkcija koja će sva slova ukoliko upišemo kraj malim uvećati i program će prestati sa radom. Kroz if- else petlju smo odradili unos godina koje se dalje provjeravaju kroz if-else petlje koji takmičar pripada kojoj kategoriji, else služi da ispiše ukoliko je unešen neki drugi karakter umjesto cijelog broja i traži ponovni unos.

7. ZADATAK 6

Napisati C# program koji od korisnika čitava cijeli broj $n \geq 3$, potom niz od n realnih brojeva za koje se zna da su svi različiti (i to ne treba provjeravati) i onda ispisuje učitani niz, ali uz preskakanje najmanjeg i najvećeg elementa u nizu.

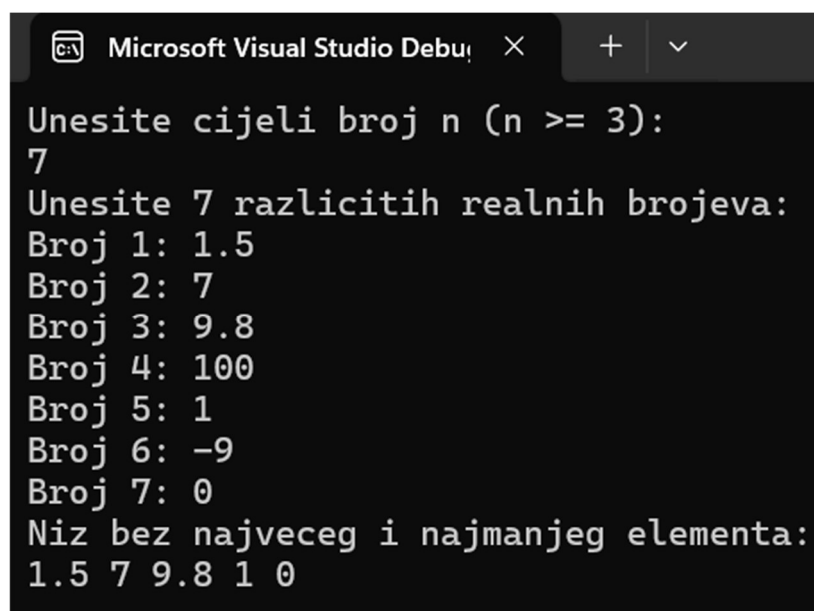
7.1. IZRADA ZADATKA 6

Slika 12. Zadatak 6



```
1  class Program
2  {
3      static void Main()
4      {
5          Console.WriteLine("Unesite cijeli broj n (n >= 3):");
6          int n;
7          while (true)
8          {
9              if (int.TryParse(Console.ReadLine(), out n) && n >= 3)
10             {
11                 break;
12             }
13             Console.WriteLine("Greška! Unesite cijeli broj n koji je veći ili jednak 3:");
14         }
15         double[] niz = new double[n];
16         Console.WriteLine($"Unesite {n} različitih realnih brojeva:");
17         for (int i = 0; i < n; i++)
18         {
19             while (true)
20             {
21                 Console.Write($"Broj {i + 1}: ");
22                 if (double.TryParse(Console.ReadLine(), out niz[i]))
23                 {
24                     break;
25                 }
26                 Console.WriteLine("Greška! Unesite ispravan realan broj.");
27             }
28         }
29         double max = niz.Max();
30         double min = niz.Min();
31         Console.WriteLine("Niz bez najvećeg i najmanjeg elementa:");
32         foreach (double broj in niz)
33         {
34             if (broj != max && broj != min)
35             {
36                 Console.Write(broj + " ");
37             }
38         }
39         Console.WriteLine();
40     }
41 }
```

Slika 13. Output zadatka 6

The image shows a screenshot of the Microsoft Visual Studio Debug Console. The window title is "Microsoft Visual Studio Debug Console". The output text is as follows:

```
Unesite cijeli broj n (n >= 3):  
7  
Unesite 7 razlicitih realnih brojeva:  
Broj 1: 1.5  
Broj 2: 7  
Broj 3: 9.8  
Broj 4: 100  
Broj 5: 1  
Broj 6: -9  
Broj 7: 0  
Niz bez najveceg i najmanjeg elementa:  
1.5 7 9.8 1 0
```

7.2. OBRAZLOŽENJE ZADATKA 6

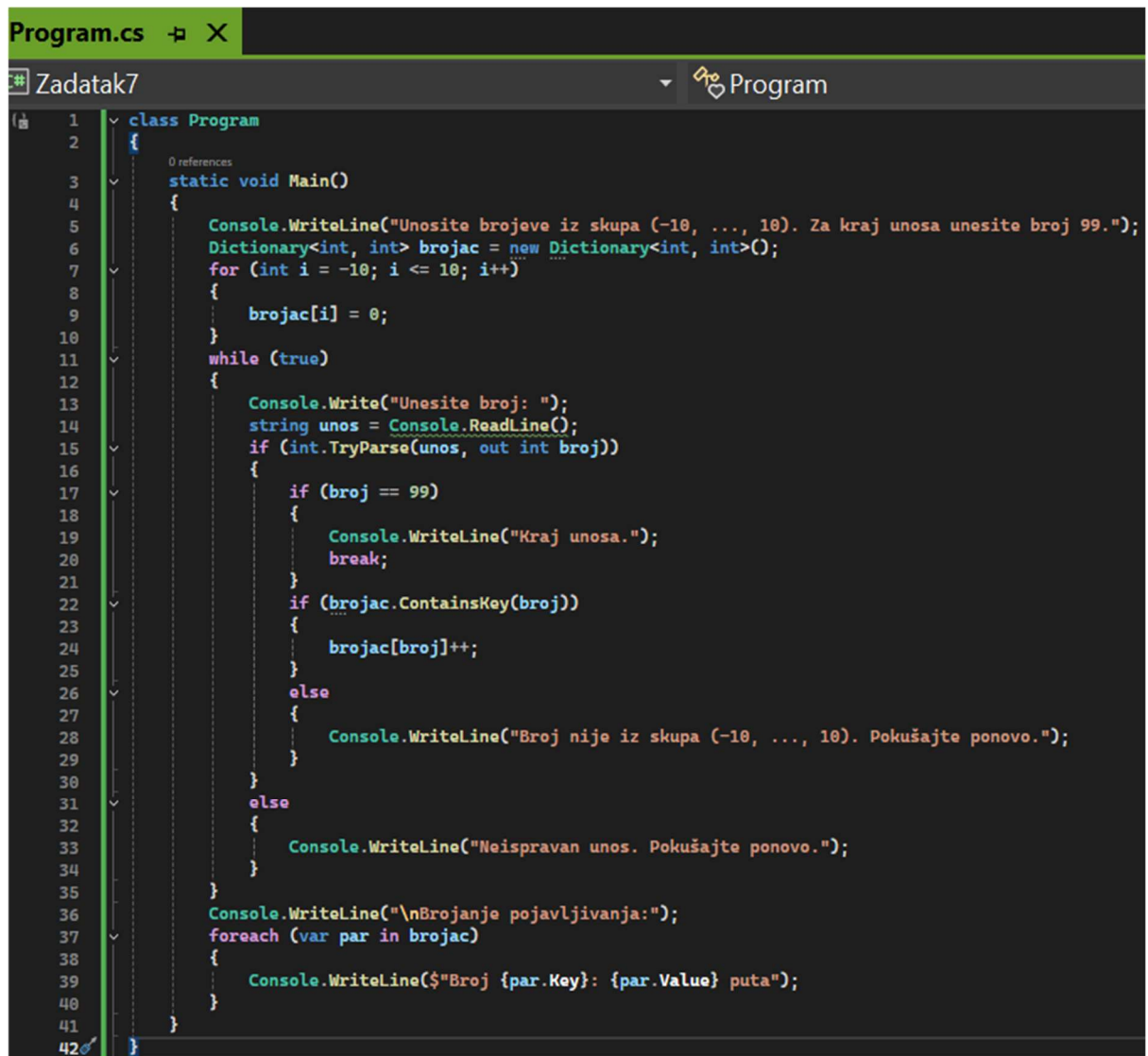
Program omogućava unos niza realnih brojeva i pronalazi najveći i najmanji element u tom nizu i ispisuje niz bez tih elemenata. Prvo program traži da korisnik unese koliku dužinu niza želi s tim da taj broj mora biti veći ili jednak tri. Koristi se while petlja kako bi se provjerilo da je unos validan i metoda `int.TryParse` da konvertuje u cijeli broj. Kada je uslov validan petlja se prekida ukoliko nije petlja će se ponavljati dok ne bude uslov zadovoljen. U for petlji korisnik unosi n realnih brojeva koristi se `double.TryParse` kako bi se provjerila validnost unosa, ako nije validan ispisuje se greška i traži se ponovni unos. Na kraju petlje svi validni brojevi su smješteni u niz, potom se traži najveći i najmanji element u nizu uz pomoć metoda `Max()` i `Min()`. Zatim se treba ispisati niz bez najvećeg i najmanjeg elementa u nizu. Za to koristimo foreach petlju za iteraciju kroz niz, gdje se svaki element koji nije jednak min i max ispisuje, uvjet u petlji osigurava da se preskoče najmanji i najveći element u nizu.

8. ZADATAK 7

Napisati C# program koji od korisnika učitava brojeve iz skupa {-10, ..., -1, 0, 1, ..., 10} i broji koliko se puta svaki od njih pojavio. Kraj unosa označen je brojem 99 koji se ne računa.

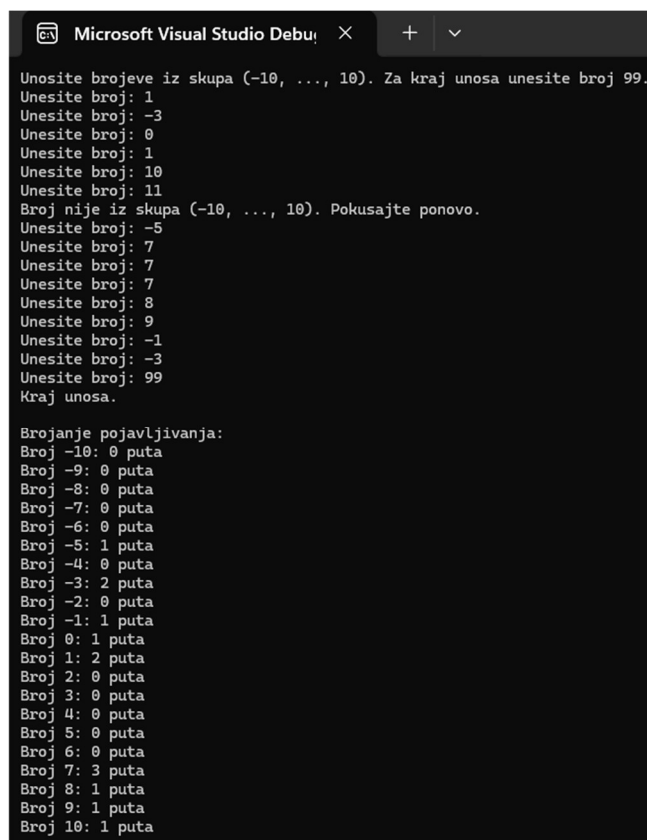
8.1. IZRADA ZADATKA 7

Slika 14. Zadatak 7



```
Program.cs  + X
# Zadatak7  Program
1  class Program
2  {
3      0 references
4      static void Main()
5      {
6          Console.WriteLine("Unesite brojeve iz skupa (-10, ..., 10). Za kraj unosa unesite broj 99.");
7          Dictionary<int, int> brojac = new Dictionary<int, int>();
8          for (int i = -10; i <= 10; i++)
9          {
10             brojac[i] = 0;
11         }
12         while (true)
13         {
14             Console.Write("Unesite broj: ");
15             string unos = Console.ReadLine();
16             if (int.TryParse(unos, out int broj))
17             {
18                 if (broj == 99)
19                 {
20                     Console.WriteLine("Kraj unosa.");
21                     break;
22                 }
23                 if (brojac.ContainsKey(broj))
24                 {
25                     brojac[broj]++;
26                 }
27                 else
28                 {
29                     Console.WriteLine("Broj nije iz skupa (-10, ..., 10). Pokušajte ponovo.");
30                 }
31             }
32             else
33             {
34                 Console.WriteLine("Neispravan unos. Pokušajte ponovo.");
35             }
36         }
37         Console.WriteLine("\nBrojanje pojavljivanja:");
38         foreach (var par in brojac)
39         {
40             Console.WriteLine($"Broj {par.Key}: {par.Value} puta");
41         }
42     }
}
```


Slika 15. Output zadatka 7



```
Microsoft Visual Studio Debug Console
Unosite brojeve iz skupa (-10, ..., 10). Za kraj unosa unesite broj 99.
Unosite broj: 1
Unosite broj: -3
Unosite broj: 0
Unosite broj: 1
Unosite broj: 10
Unosite broj: 11
Broj nije iz skupa (-10, ..., 10). Pokušajte ponovo.
Unosite broj: -5
Unosite broj: 7
Unosite broj: 7
Unosite broj: 7
Unosite broj: 8
Unosite broj: 9
Unosite broj: -1
Unosite broj: -3
Unosite broj: 99
Kraj unosa.

Brojanje pojavljivanja:
Broj -10: 0 puta
Broj -9: 0 puta
Broj -8: 0 puta
Broj -7: 0 puta
Broj -6: 0 puta
Broj -5: 1 puta
Broj -4: 0 puta
Broj -3: 2 puta
Broj -2: 0 puta
Broj -1: 1 puta
Broj 0: 1 puta
Broj 1: 2 puta
Broj 2: 0 puta
Broj 3: 0 puta
Broj 4: 0 puta
Broj 5: 0 puta
Broj 6: 0 puta
Broj 7: 3 puta
Broj 8: 1 puta
Broj 9: 1 puta
Broj 10: 1 puta
```

8.2. OBRAZLOŽENJE ZADATKA 7

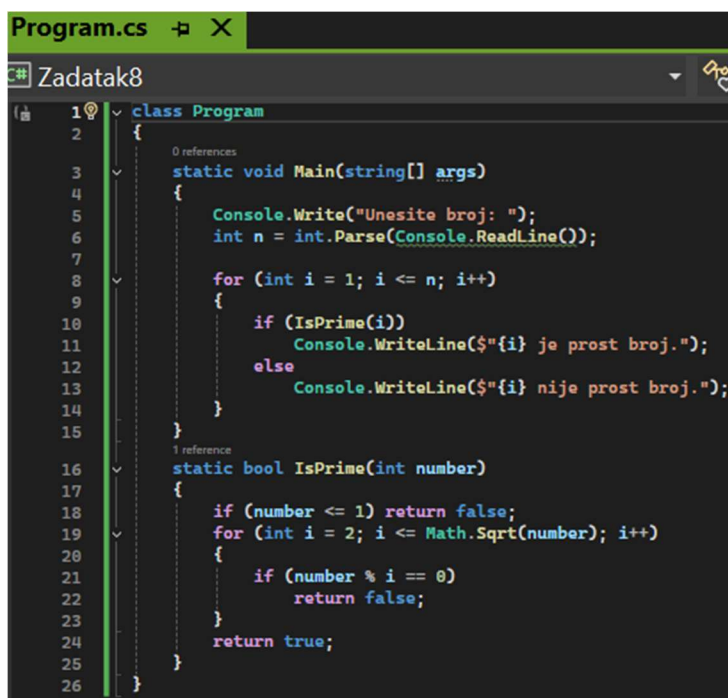
Program omogućava da korisnik unese brojeve iz skupa $\{-10, 10\}$, te prebroji koliko puta se neki broj ponavlja, unos se prekida upisom broja 99 i ispisuje se koliko puta se svaki broj pojavio u nizu. Prvo što uradimo je da definišemo skup brojeva pomoću for petlje i on se koristi za inicijalizaciju rječnika (dictionary) koja se postavlja na 0 i predstavlja koliko se neki broj ponavlja. Koristimo while petlju za unos brojeva, izvršava se sve dok ne unesemo broj 99. Sa TryParse provjerava jel unos validan cijeli broj, potom se provjerava da li broj pripada skupu, ako pripada broj se dodaje u rječnik i brojač se povećava za 1, ukoliko ne pripada ispisuje se greška i unos se ponavlja. Ukoliko unos nije ispravan unesen neki drugi karakter umjesto cijelog broja ispisuje se greška i traži se ponovni unos. Unosom 99 program se prekida i ispisuje se koliko puta se neki broj pojavljuje u programu. `par.Key` predstavlja broj, a `par.Value` predstavlja broj ponavljanja.

9. ZADATAK 8

Napišite program koji ispisuje sve brojeve od unijetog broja i za svaki broj određuje da li je prost ili nije. Treba se implementirati metoda za provjeru prostih brojeva.

9.1. IZRADA ZADATKA 8

Slika 16. Zadatak 8



```
Program.cs X
Zadatak8
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.Write("Unesite broj: ");
6         int n = int.Parse(Console.ReadLine());
7
8         for (int i = 1; i <= n; i++)
9         {
10             if (IsPrime(i))
11                 Console.WriteLine($"{i} je prost broj.");
12             else
13                 Console.WriteLine($"{i} nije prost broj.");
14         }
15     }
16     static bool IsPrime(int number)
17     {
18         if (number <= 1) return false;
19         for (int i = 2; i <= Math.Sqrt(number); i++)
20         {
21             if (number % i == 0)
22                 return false;
23         }
24         return true;
25     }
26 }
```

Slika 17. Output zadatka 8



```
Microsoft Visual Studio Debug Console
Unesite broj: 10
1 nije prost broj.
2 je prost broj.
3 je prost broj.
4 nije prost broj.
5 je prost broj.
6 nije prost broj.
7 je prost broj.
8 nije prost broj.
9 nije prost broj.
10 nije prost broj.
```

9.2. OBRAZLOŽENJE ZADATKA 8

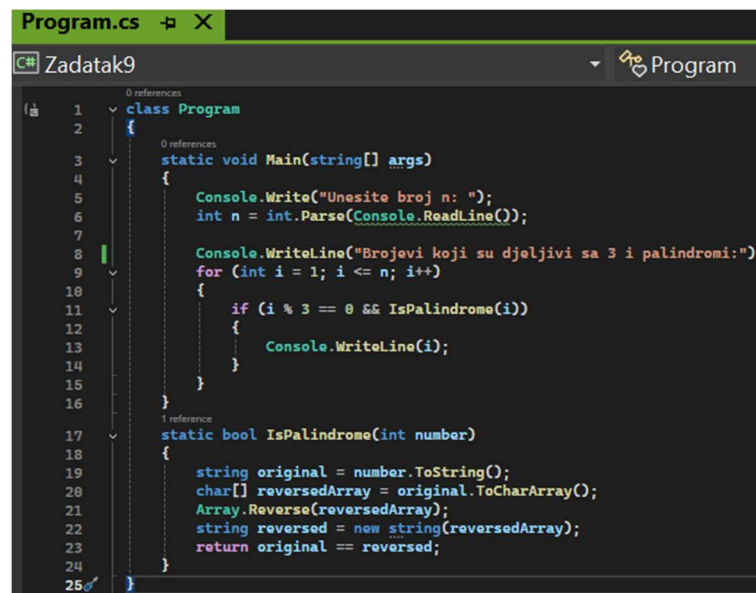
Prost broj je broj koji je djeljiv samo sam sa sobom i sa 1. Program omogućava korisniku unos cijelog broja za sve brojeve od 1 pa do tog unesenog broja određuje da li su brojevi prosti ili nisu. Program traži da korisnik unese cijeli broj koji se kovertuje u int, ukoliko je nevalidna vrijednost ispisat će grešku. For petlju koristimo za provjeru svih brojeva, za svaki broj i poziva se metoda IsPrime() kako bi se provjerilo da li je broj prost, ako metoda vrati true ispisuje se da je broj prost, ukoliko je false ispisuje se da nije prost. U metodi za provjeru prostog broje prvo eliminišemo brojeve manje ili jednake 1 i provjeravamo djeljivost brojeva od 2 pa na dalje. Ako je broj djeljiv sa bilo kojim brojem vraća se false i broj nije prost, ako nije pronađen nijedan djelilac broj je prost.

10. ZADATAK 9

Napišite program koji pronalazi sve brojeve od 1 do unijetog broja n, koji su istovremeno djeljivi sa 3 i palindromi (broj koji je isti kad se čita s obe strane).

10.1. IZRADA ZADATKA 9

Slika 18. Zadatak 9

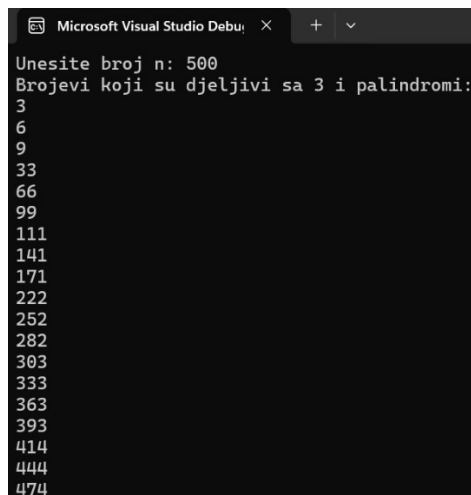


```
Program.cs
Zadatak9
class Program
{
    static void Main(string[] args)
    {
        Console.Write("Unesite broj n: ");
        int n = int.Parse(Console.ReadLine());

        Console.WriteLine("Brojevi koji su djeljivi sa 3 i palindromi:");
        for (int i = 1; i <= n; i++)
        {
            if (i % 3 == 0 && IsPalindrome(i))
            {
                Console.WriteLine(i);
            }
        }
    }

    static bool IsPalindrome(int number)
    {
        string original = number.ToString();
        char[] reversedArray = original.ToCharArray();
        Array.Reverse(reversedArray);
        string reversed = new string(reversedArray);
        return original == reversed;
    }
}
```

Slika 19. Output zadatka 9



```
Microsoft Visual Studio Debug Console
Unesite broj n: 500
Brojevi koji su djeljivi sa 3 i palindromi:
3
6
9
33
66
99
111
141
171
222
252
282
303
333
363
393
414
444
474
```

10.2. OBRAZLOŽENJE ZADATKA 9

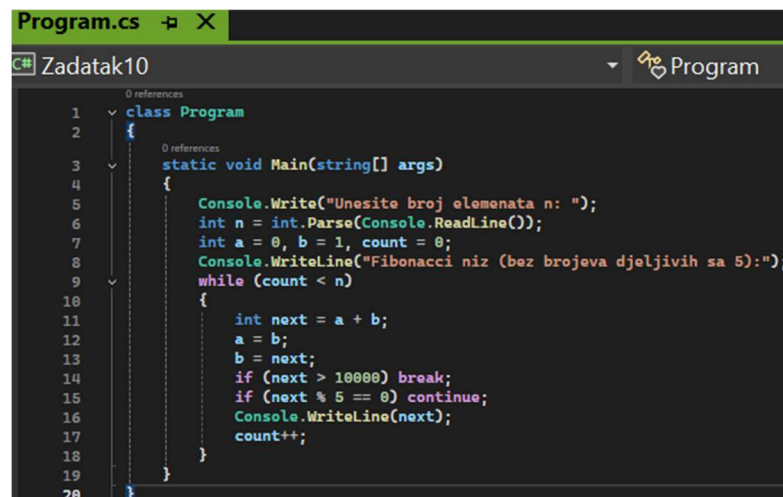
Program ispisuje sve brojeve od 1 pa do unesenog broja koji su djeljivi sa 3 i istovremeno palindromi. Kada unesemo cijeli broj uneseni string se pretvara u int. For petlja služi da iterira sve brojeve od 1 pa do unesenog broja. Kroz if petlju se provjerava djeljivost sa 3 to jest provjerava se da li je ostatak pri dijeljenju sa 3 jednak 0 i provjerava se dali je broj palindrom pozivanjem metode IsPalindrome(), ako broj ispunjava uslov broj se ispisuje. U metodi IsPalindrome() prvo pretvaramo broj u string radi lakše obrade. String original se pretvara u niz karaktera. Funkcija Array.Reverse obrće niz karaktera. Potom se obrnuti niz pretvara u string i zadnje preostaje poređenje originala i obrnutog stringa ukoliko metoda vraća true broj je palindrom ukoliko nije vraća false. Kada se ispune oba uslova broj se ispisuje na ekran.

11. ZADATAK 10

Implementirati program koji generiše prvih n brojeva Fibonacci niza, ali da se preskaču svi brojevi koji su djeljivi sa 5 i generisanje da se završi ukoliko broj prelazi 10 000.

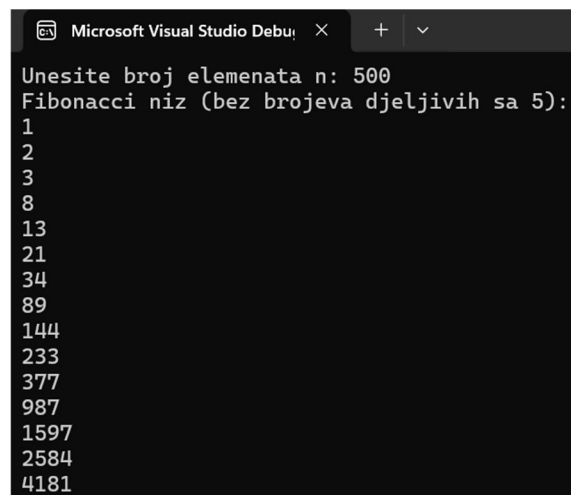
11.1. IZRADA ZADATKA 10

Slika 20. Zadatak 10



```
Program.cs X
Zadatak10 Program
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.Write("Unesite broj elemenata n: ");
6         int n = int.Parse(Console.ReadLine());
7         int a = 0, b = 1, count = 0;
8         Console.WriteLine("Fibonacci niz (bez brojeva djeljivih sa 5):");
9         while (count < n)
10         {
11             int next = a + b;
12             a = b;
13             b = next;
14             if (next > 10000) break;
15             if (next % 5 == 0) continue;
16             Console.WriteLine(next);
17             count++;
18         }
19     }
20 }
```

Slika 21. Output zadatka 10



```
Microsoft Visual Studio Debug Console
Unesite broj elemenata n: 500
Fibonacci niz (bez brojeva djeljivih sa 5):
1
2
3
8
13
21
34
89
144
233
377
987
1597
2584
4181
```

11.2. OBRAZLOŽENJE ZADATKA 10

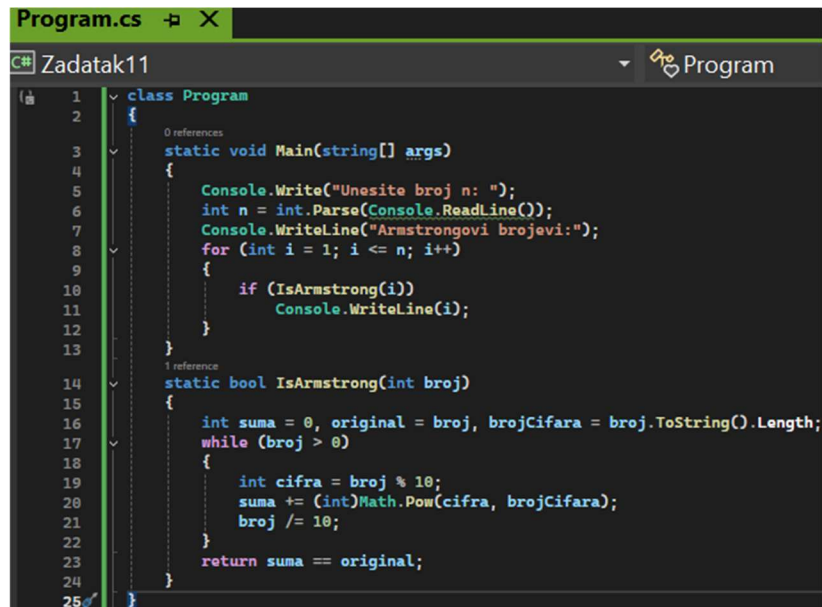
Program generiše Fibonacci niz i ispisuje sve elemente osim onih koji su djeljivi sa 5. Korisnik unosi broj koliko će se elemenata Fibonacci niza ispisati, a i b su prve dvije vrijednosti u nizu count prati koliko elemenata je ispisano. Fibonacci niz se generiše tako što je svaki broj suma prethodna dva broja, nakon generisanja broja ažuriraju se vrijednosti a i b za sljedeću iteraciju. Ako novo generisani broj premaši vrijednost 10 000 petlja se prekida i dalje se brojevi ne ispisuju. Potom se provjerava djeljivost sa 5 ukoliko je ostatak pri dijeljenju sa 5 jednak 0 taj broj se preskače, continue naredba vraća kontrolu na početak petlje, i broj koji je djeljiv s 5 neće biti ispisan niti dodan u brojanje. Ako broj nije djeljiv sa 5 i nije veći od 10 000 ispisuje se na ekran, count promjenljiva prati broj ispisanih elemenata i uvećava se za 1.

12. ZADATAK 11

Napisati program koji pronalazi sve Armstrongove brojeve u opsegu od 1 pa do unesenog broja n. Armstrongov broj je broj jednak zbiru svojih cifara podignutih na stepen broja cifara (npr. $153 = 1^3 + 5^3 + 3^3$)

12.1. IZRADA ZADATKA 11

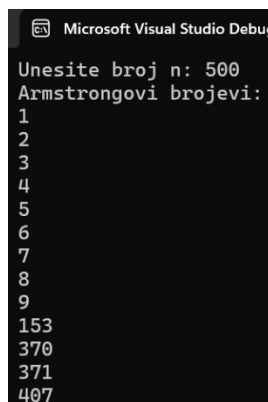
Slika 22. Zadatak 11



```
Program.cs
Zadatak11
class Program
{
    static void Main(string[] args)
    {
        Console.Write("Unesite broj n: ");
        int n = int.Parse(Console.ReadLine());
        Console.WriteLine("Armstrongovi brojevi:");
        for (int i = 1; i <= n; i++)
        {
            if (IsArmstrong(i))
                Console.WriteLine(i);
        }
    }

    static bool IsArmstrong(int broj)
    {
        int suma = 0, original = broj, brojCifara = broj.ToString().Length;
        while (broj > 0)
        {
            int cifra = broj % 10;
            suma += (int)Math.Pow(cifra, brojCifara);
            broj /= 10;
        }
        return suma == original;
    }
}
```

Slika 23. Output zadatka 11



```
Microsoft Visual Studio Debug Console
Unesite broj n: 500
Armstrongovi brojevi:
1
2
3
4
5
6
7
8
9
153
370
371
407
```


12.2. OBRAZLOŽENJE ZADATKA 11

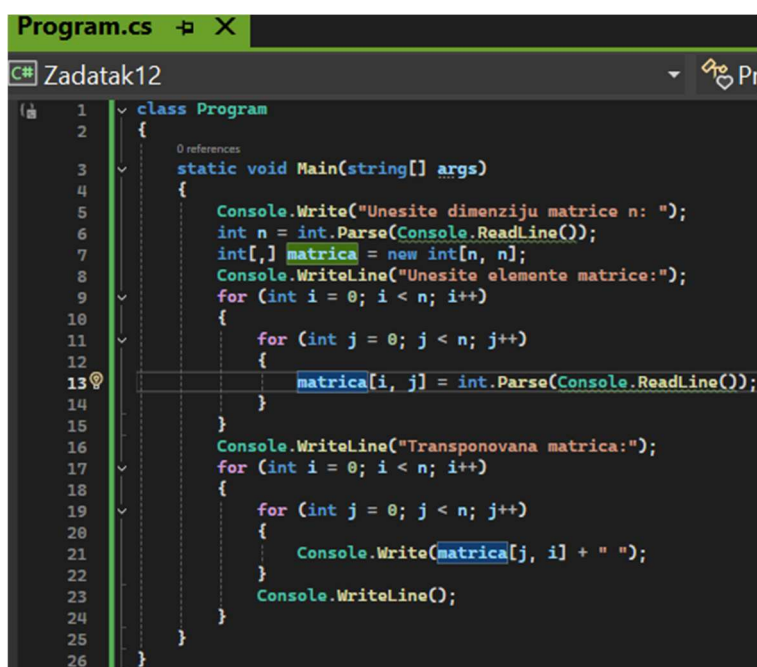
Program pronalazi i ispisuje Armstrongove brojeve u rasponu od 1 pa do n koje korisnik unosi. Prvo se unosi broj do kojeg želimo da provjerimo Armstrongove brojeve, te konvertuje se tekst u int. Potom se ispisuju Armstrongovi brojevi kroz for petlju koja iterira sve brojeve od 1 pa do n, funkcija IsArmstrong provjerava da li je broj Armstrongov, ako jeste ispisuje se na konzoli. Što se tiče funkcije IsArmstrong prvo inicijalizujemo sumu koja služi za skladištenje zbira cifara podignutih na odgovarajuću potenciju, original pohranjuje originalni broj za kasnije poređenje, brojCifara određuje broj cifara u broju. Pomoću broj%10 ekstraktujemo zadnju cifru broja, sa Math.Pow podižemo cifru na potenciju jednaku broju cifara u originalnom broju, a broj/=10 uklanjamo zadnju cifru. Zadnje se provjerava suma da li je jednaka originalnom broju, ako vraća true ispisuje se na konzolu.

13. ZADATAK 12

Napišite program koji unosi kvadratnu matricu $n \times n$, a zatim ispisuje njen transponovani oblik.

13.1. IZRADA ZADATAKA 12

Slika 24. Zadatak 12



```
Program.cs X
C# Zadatak12
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.Write("Unesite dimenziju matrice n: ");
6         int n = int.Parse(Console.ReadLine());
7         int[,] matrica = new int[n, n];
8         Console.WriteLine("Unesite elemente matrice:");
9         for (int i = 0; i < n; i++)
10         {
11             for (int j = 0; j < n; j++)
12             {
13                 matrica[i, j] = int.Parse(Console.ReadLine());
14             }
15         }
16         Console.WriteLine("Transponovana matrica:");
17         for (int i = 0; i < n; i++)
18         {
19             for (int j = 0; j < n; j++)
20             {
21                 Console.Write(matrica[j, i] + " ");
22             }
23             Console.WriteLine();
24         }
25     }
26 }
```

Slika 25. Output zadatka 12



```
Microsoft Visual Studio Debu: X +
Unesite dimenziju matrice n: 3
Unesite elemente matrice:
1
2
3
4
5
6
7
8
9
Transponovana matrica:
1 4 7
2 5 8
3 6 9
```

13.2. OBRAZLOŽENJE ZADATKA 12

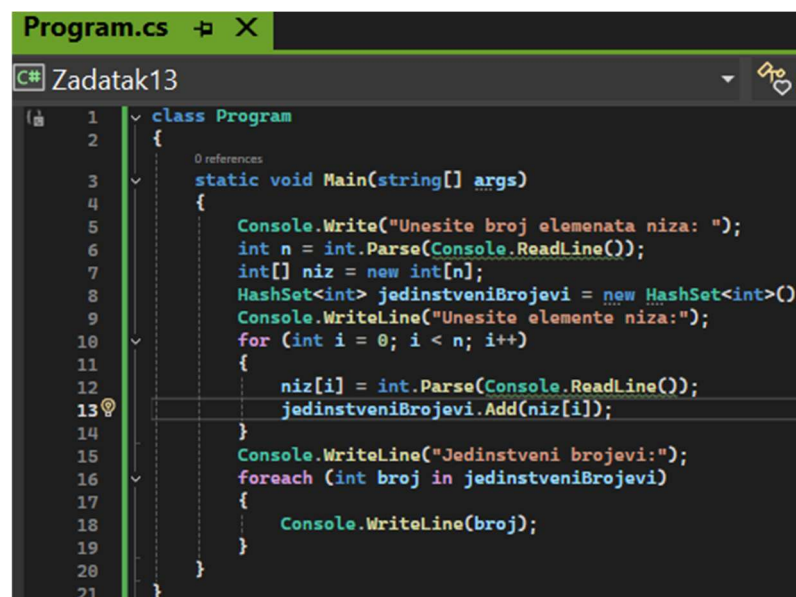
Program omogućuje unos kvadratne matrice dimenzije $n \times n$, zatim računa i ispisuje transponovanu matricu, ona podrazumijeva zamjenu redova i kolona. Prvo unosimo dimenzije matrice, te se konvertuje tekst u int, potom deklariramo matricu `int[,]` matrica i elementi matrice su tipa `int`. Potom se unose elementi matrice, vanjska for petlja predstavlja redove, unutrašnja predstavlja kolone. Te se ispisuje transponovana matrica i koristimo opet dvije ugniježdene for petlje, umjesto matrica `[i, j]` ispisuje se matrica `[j, i]`.

14. ZADATAK 13

Napisati program koji uklanja duplikate iz unijetog niza brojeva i ispisuje jedinstvene vrijednosti.

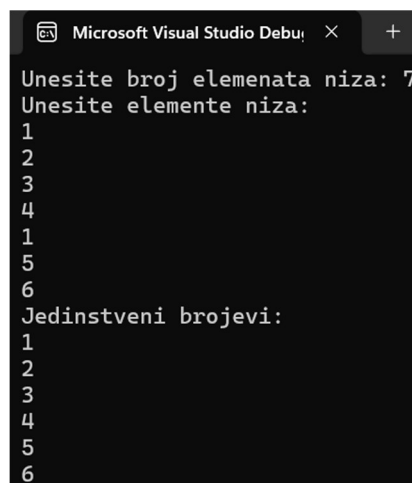
14.1. IZRADA ZADATKA 13

Slika 26. Zadatak 13



```
Program.cs X
C# Zadatak13
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.Write("Unesite broj elemenata niza: ");
6         int n = int.Parse(Console.ReadLine());
7         int[] niz = new int[n];
8         HashSet<int> jedinstveniBrojevi = new HashSet<int>();
9         Console.WriteLine("Unesite elemente niza:");
10        for (int i = 0; i < n; i++)
11        {
12            niz[i] = int.Parse(Console.ReadLine());
13            jedinstveniBrojevi.Add(niz[i]);
14        }
15        Console.WriteLine("Jedinstveni brojevi:");
16        foreach (int broj in jedinstveniBrojevi)
17        {
18            Console.WriteLine(broj);
19        }
20    }
21 }
```

Slika 27. Output zadatka 13



```
Microsoft Visual Studio Debug Console
Unesite broj elemenata niza: 7
Unesite elemente niza:
1
2
3
4
1
5
6
Jedinstveni brojevi:
1
2
3
4
5
6
```

14.2. OBRAZLOŽENJE ZADATKA 13

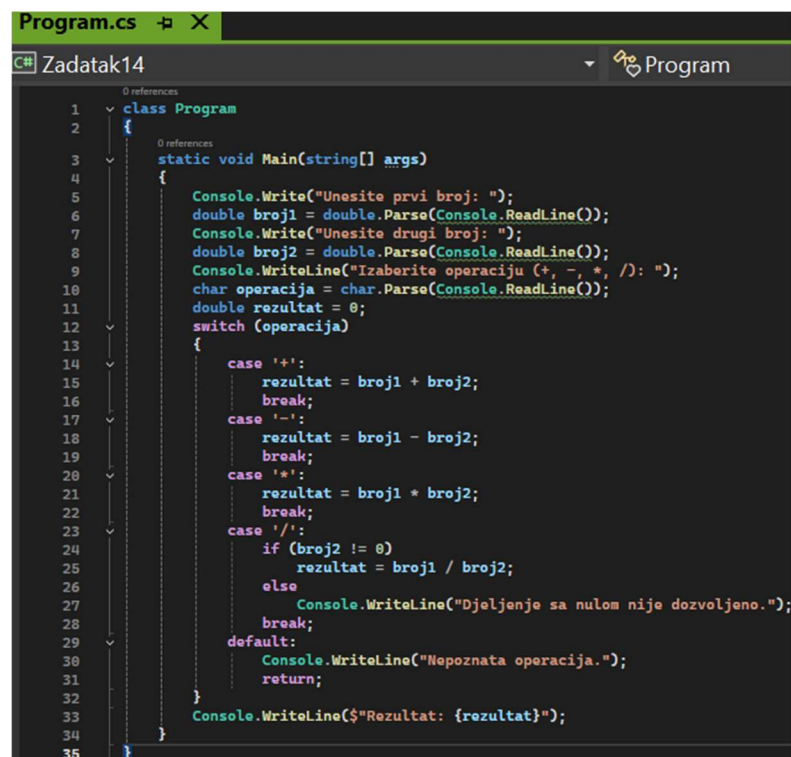
Program omogućava korisniku unos brojeva u niz, a zatim izdvaja i ispisuje jedinstvene elemente odnosno uklanja duplikate. Traži se od korisnika koliki niz će biti n i pretvara tekst u `int`. Zatim inicijalizujemo niz dimenzija koje unesemo na početku. `HashSet` je specijalna kolekcija u `C#` koja čuva samo jedinstvene elemente i ne dozvoljava dupliciranje elemenata. Kroz `for` petlju je omogućen unos elementata niza n . Unosimo element u niz koji se čuva u odgovarajućem indeksu niza, element iz niza se dodaje u `HashSet` kolekciju ako postoji u `HashSetu`-u on se neće dodati. Potom se ispisuju jedinstveni elementi uz pomoć `foreach` petlje koja prolazi kroz sve elemente u `HashSetu`.

15. ZADATAK 14

Napisati program koji omogućava korisniku da unese dva broja i izabere jednu od osnovnih matematičkih operacija (+, -, *, /), zatim izračunava i prikazuje rezultat.

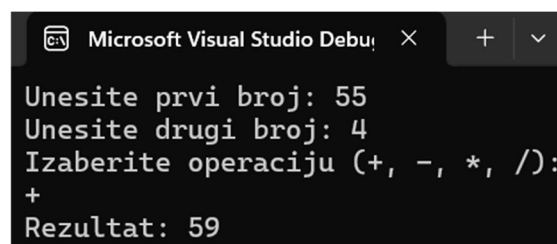
15.1. IZRADA ZADATKA 14

Slika 28. Zadatak 14



```
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.WriteLine("Unesite prvi broj: ");
6         double broj1 = double.Parse(Console.ReadLine());
7         Console.WriteLine("Unesite drugi broj: ");
8         double broj2 = double.Parse(Console.ReadLine());
9         Console.WriteLine("Izaberite operaciju (+, -, *, /): ");
10        char operacija = char.Parse(Console.ReadLine());
11        double rezultat = 0;
12        switch (operacija)
13        {
14            case '+':
15                rezultat = broj1 + broj2;
16                break;
17            case '-':
18                rezultat = broj1 - broj2;
19                break;
20            case '*':
21                rezultat = broj1 * broj2;
22                break;
23            case '/':
24                if (broj2 != 0)
25                    rezultat = broj1 / broj2;
26                else
27                    Console.WriteLine("Djeljenje sa nulom nije dozvoljeno.");
28                break;
29            default:
30                Console.WriteLine("Nepoznata operacija.");
31                return;
32        }
33        Console.WriteLine($"Rezultat: {rezultat}");
34    }
35 }
```

Slika 29. Output zadatka 14



```
Microsoft Visual Studio Debu
Unesite prvi broj: 55
Unesite drugi broj: 4
Izaberite operaciju (+, -, *, /):
+
Rezultat: 59
```

15.2. OBRAZLOŽENJE ZADATKA 14

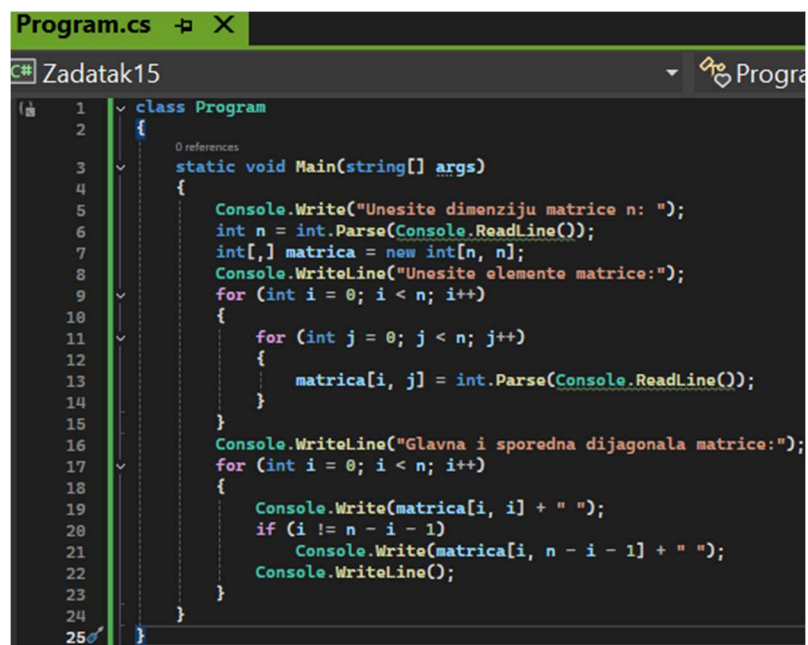
Program implementira jednostavan kalkulator koji omogućava korisniku da unese dva broja, odabere matematičku operaciju i da dobije rezultat. Prvo se ispisuje korisniku da unese broj koji je u obliku stringa i pretvara se u double. Traži se potom da se izabere jedna od četiri moguće operacije i pretvara se u char. Rezultat se inicijalizuje na 0 i koristi se za čuvanje rezultata matematičke operacije. Pomoću switch bloka se provjerava koja operacija je unijeta i izvršava se odgovarajući case. Kod dijeljenja provjeravamo pomoću if-else da broj2 nije 0.

16. ZADATAK 15

Napisati program koji unosi kvadratnu matricu $n \times n$, a zatim ispisuje samo glavnu i sporednu dijagonalu matrice.

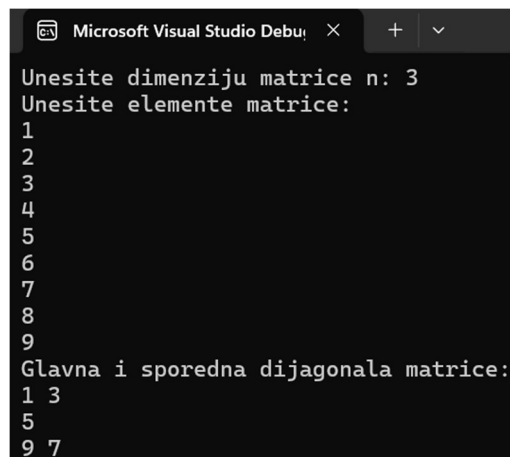
16.1. IZRADA ZADATAKA 15

Slika 30. Zadatak 15



```
Program.cs X
C# Zadatak15
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.Write("Unesite dimenziju matrice n: ");
6         int n = int.Parse(Console.ReadLine());
7         int[,] matrica = new int[n, n];
8         Console.WriteLine("Unesite elemente matrice:");
9         for (int i = 0; i < n; i++)
10         {
11             for (int j = 0; j < n; j++)
12             {
13                 matrica[i, j] = int.Parse(Console.ReadLine());
14             }
15         }
16         Console.WriteLine("Glavna i sporedna dijagonala matrice:");
17         for (int i = 0; i < n; i++)
18         {
19             Console.Write(matrica[i, i] + " ");
20             if (i != n - i - 1)
21                 Console.Write(matrica[i, n - i - 1] + " ");
22             Console.WriteLine();
23         }
24     }
25 }
```

Slika 31. Output zadatka 15



```
Microsoft Visual Studio Debug Console
Unesite dimenziju matrice n: 3
Unesite elemente matrice:
1
2
3
4
5
6
7
8
9
Glavna i sporedna dijagonala matrice:
1 3
5
9 7
```


16.2. OBRAZLOŽENJE ZADATKA 15

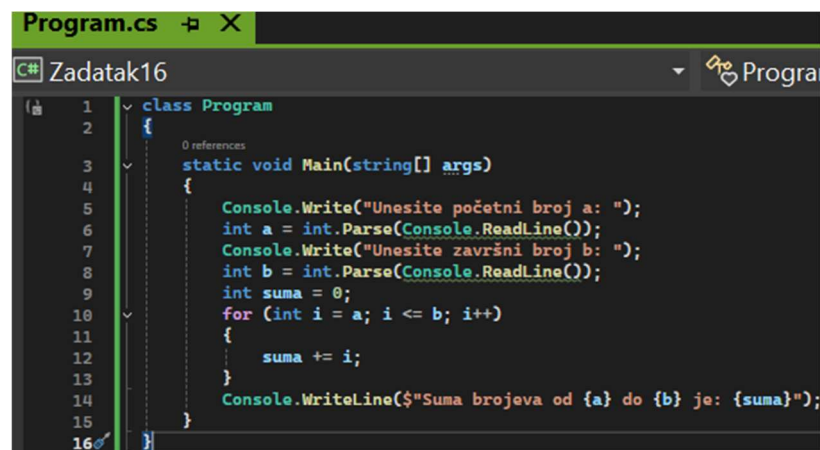
Program omogućava da se unese kvadratna matrica dimenzije $n \times n$ i da se ispisuju elementi glavne i sporedne dijagonale. Unosimo dimenziju niza i konvertujemo u int. Inicijalizujemo dvodimenzionalnu matricu `[,]`. Ugniježdene for petlje prolaze kroz svaki red i kolonu vanjska iterira redove, a unutrašnja kolone. Glavna dijagonala matrice se sastoji od elemenata za koje je indeks reda jednak indeksu kolone, sporedna dijagonala se sastoji od elemenata za koje je suma indeksa reda i kolone jednaka $n - 1$ ($i + j = n - 1$). ako se indeks reda poklapa sa indeksom sporedne dijagonale, element glavne dijagonale se ne ispisuje dva puta zbog uslova kojeg smo napisali. Na kraju nam se ispisuju elementi glavne i sporedne dijagonale.

17. ZADATAK 16

Napisati program koji računa sumu svih brojeva u intervalu od a do b, gdje korisnik unosi vrijednosti za a i b.

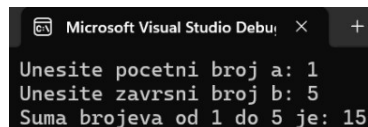
17.1. IZRADA ZADATKA 16

Slika 32. Zadatak 16



```
Program.cs X
C# Zadatak16
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.Write("Unesite početni broj a: ");
6         int a = int.Parse(Console.ReadLine());
7         Console.Write("Unesite završni broj b: ");
8         int b = int.Parse(Console.ReadLine());
9         int suma = 0;
10        for (int i = a; i <= b; i++)
11        {
12            suma += i;
13        }
14        Console.WriteLine($"Suma brojeva od {a} do {b} je: {suma}");
15    }
16 }
```

Slika 33. Output zadatka 16



```
Microsoft Visual Studio Debu: X +
Unesite pocetni broj a: 1
Unesite zavrnsni broj b: 5
Suma brojeva od 1 do 5 je: 15
```

17.2. OBRAZLOŽENJE ZADATKA 16

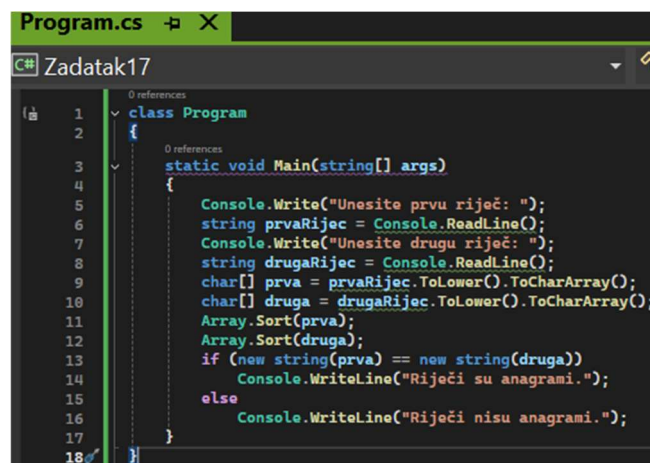
Program računa sumu svih brojeva u opsegu između dva unesena broja uključujući i krajeve opsega. Prvo unosimo početni i krajnji broj koji se konvertuju u int, suma se inicijalizuje na 0 i čuvat će ukupnu sumu svih brojeva u opsegu. Kroz for petlju računamo sumu, početnu vrijednost iteratora i stavljamo da je a, i petlja se izvršava do $i \leq b$, na kraju računamo sumu, trenutna vrijednost se dodaje na sumu.

18. ZADATAK 17

Napisati program koji provjerava da li su dvije riječi anagrami (sadrže iste karaktere u različitom redoslijedu).

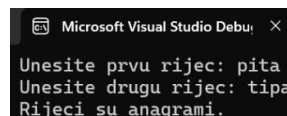
18.1. IZRADA ZADATKA 17

Slika 34. Zadatak 17



```
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Console.WriteLine("Unesite prvu riječ: ");
6         string prvaRijec = Console.ReadLine();
7         Console.WriteLine("Unesite drugu riječ: ");
8         string drugaRijec = Console.ReadLine();
9         char[] prva = prvaRijec.ToLower().ToCharArray();
10        char[] druga = drugaRijec.ToLower().ToCharArray();
11        Array.Sort(prva);
12        Array.Sort(druga);
13        if (new string(prva) == new string(druga))
14            Console.WriteLine("Riječi su anagrami.");
15        else
16            Console.WriteLine("Riječi nisu anagrami.");
17    }
18 }
```

Slika 35. Output zadatka 17



```
Microsoft Visual Studio Debug Console
Unesite prvu riječ: pita
Unesite drugu riječ: tipa
Riječi su anagrami.
```

18.2. OBRAZLOŽENJE ZADATKA 17

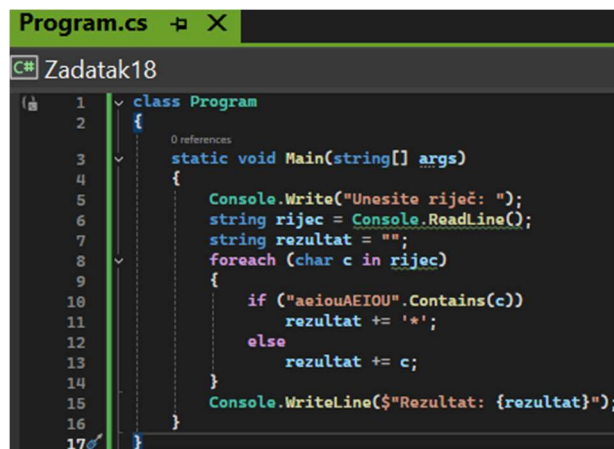
Program provjerava da li su dvije riječi anagrami. Prvo unosimo riječi, zatim ukoliko smo upisali neku riječ velikim slovima smanjimo ih sa `ToLowerCase()`, a sa `ToCharArray()` stringove pretvaramo u karaktere za lakšu manipulaciju. `Array.Sort()` sortira niz karaktera u rastućem redoslijedu, ako su riječi anagrami nizovi će biti isti. Kroz if-else petlju poredimo nizove karaktera, sa `new string` pretvara niz karaktera nazad u string, ako su stringovi jednaki riječi su anagrami, ukoliko nisu ispisuje se da riječi nisu anagrami.

19. ZADATAK 18

Napisati program koji zamjenjuje sve samoglasnike u unijetoj riječi sa znakom *.

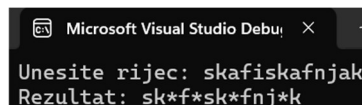
19.1. IZRADA ZADATKA 18

Slika 36. Zadatak 18



```
Program.cs  X
C# Zadatak18
1  class Program
2  {
3      static void Main(string[] args)
4      {
5          Console.Write("Unesite riječ: ");
6          string rijec = Console.ReadLine();
7          string rezultat = "";
8          foreach (char c in rijec)
9          {
10             if ("aeiouAEIOU".Contains(c))
11                 rezultat += '*';
12             else
13                 rezultat += c;
14             }
15             Console.WriteLine($"Rezultat: {rezultat}");
16         }
17     }
```

Slika 37. Output zadatka 18



```
Microsoft Visual Studio Debug Console
Unesite rijec: skafiskafnjak
Rezultat: sk*f*sk*f*nj*k
```

19.2. OBRAZLOŽENJE ZADATKA 18

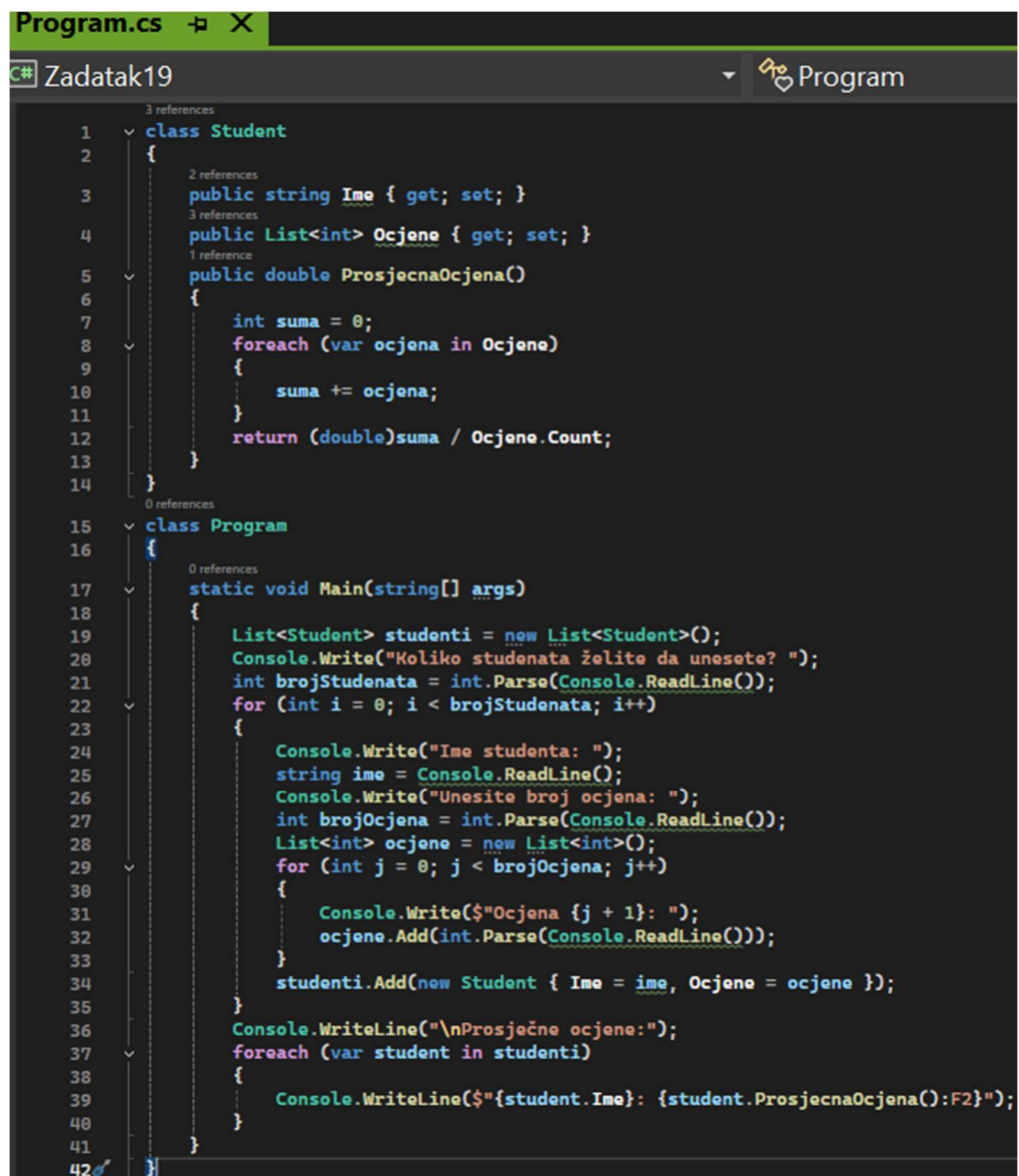
Program omogućava da korisnik unese riječ, a zatim zamijeni sve samoglasnike u riječi sa *. Prvo unosimo riječ i ona se čuva kao string. Rezultat je promjenjiva koja će čuvati izmijenjenu verziju unijete riječi. Petlja foreach prolazi kroz svaki karakter tipa char u stringu riječ, svaki karakter riječi se dodjeljuje promjenljivoj c u svakoj iteraciji. U if-else petlji se provjerava dali je karakter c jedan od samoglasnika, .Contains(c) je metoda koja provjerava da li je karakter c prisutan u stringu, ako je c samoglasnik u rezultat dodajemo *, u else ako karakter nije samoglasnik samo ga dodajemo u rezultat bez promjene. Na kraju se ispisuje rezultat gdje su samoglasnici izmijenjeni *.

20. ZADATAK 19

Napisati program koji omogućava unos ocjena za više studenata i računa prosječnu ocjenu svakog studenta.

20.1. IZRADA ZADATKA 19

Slika 38. Zadatak 19

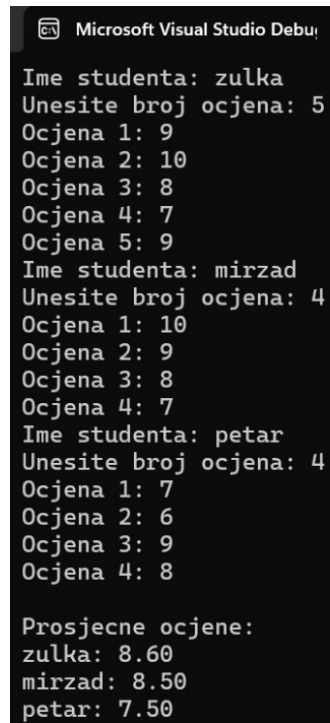


```
Program.cs X
Zadatak19 Program

1  class Student
2  {
3      public string Ime { get; set; }
4      public List<int> Ocjene { get; set; }
5      public double ProsjecnaOcjena()
6      {
7          int suma = 0;
8          foreach (var ocjena in Ocjene)
9          {
10             suma += ocjena;
11          }
12          return (double)suma / Ocjene.Count;
13      }
14  }

15  class Program
16  {
17      static void Main(string[] args)
18      {
19          List<Student> studenti = new List<Student>();
20          Console.Write("Koliko studenata želite da unesete? ");
21          int brojStudenata = int.Parse(Console.ReadLine());
22          for (int i = 0; i < brojStudenata; i++)
23          {
24              Console.Write("Ime studenta: ");
25              string ime = Console.ReadLine();
26              Console.Write("Unesite broj ocjena: ");
27              int brojOcjena = int.Parse(Console.ReadLine());
28              List<int> ocjene = new List<int>();
29              for (int j = 0; j < brojOcjena; j++)
30              {
31                  Console.Write($"Ocjena {j + 1}: ");
32                  ocjene.Add(int.Parse(Console.ReadLine()));
33              }
34              studenti.Add(new Student { Ime = ime, Ocjene = ocjene });
35          }
36          Console.WriteLine("\nProsječne ocjene:");
37          foreach (var student in studenti)
38          {
39              Console.WriteLine($"{student.Ime}: {student.ProsjecnaOcjena():F2}");
40          }
41      }
42  }
```

Slika 39. Output zadatka 19



```
Microsoft Visual Studio Debug Console
Ime studenta: zulka
Unesite broj ocjena: 5
Ocjena 1: 9
Ocjena 2: 10
Ocjena 3: 8
Ocjena 4: 7
Ocjena 5: 9
Ime studenta: mirzad
Unesite broj ocjena: 4
Ocjena 1: 10
Ocjena 2: 9
Ocjena 3: 8
Ocjena 4: 7
Ime studenta: petar
Unesite broj ocjena: 4
Ocjena 1: 7
Ocjena 2: 6
Ocjena 3: 9
Ocjena 4: 8

Prosječne ocjene:
zulka: 8.60
mirzad: 8.50
petar: 7.50
```

20.2. OBRAZLOŽENJE ZADATKA 19

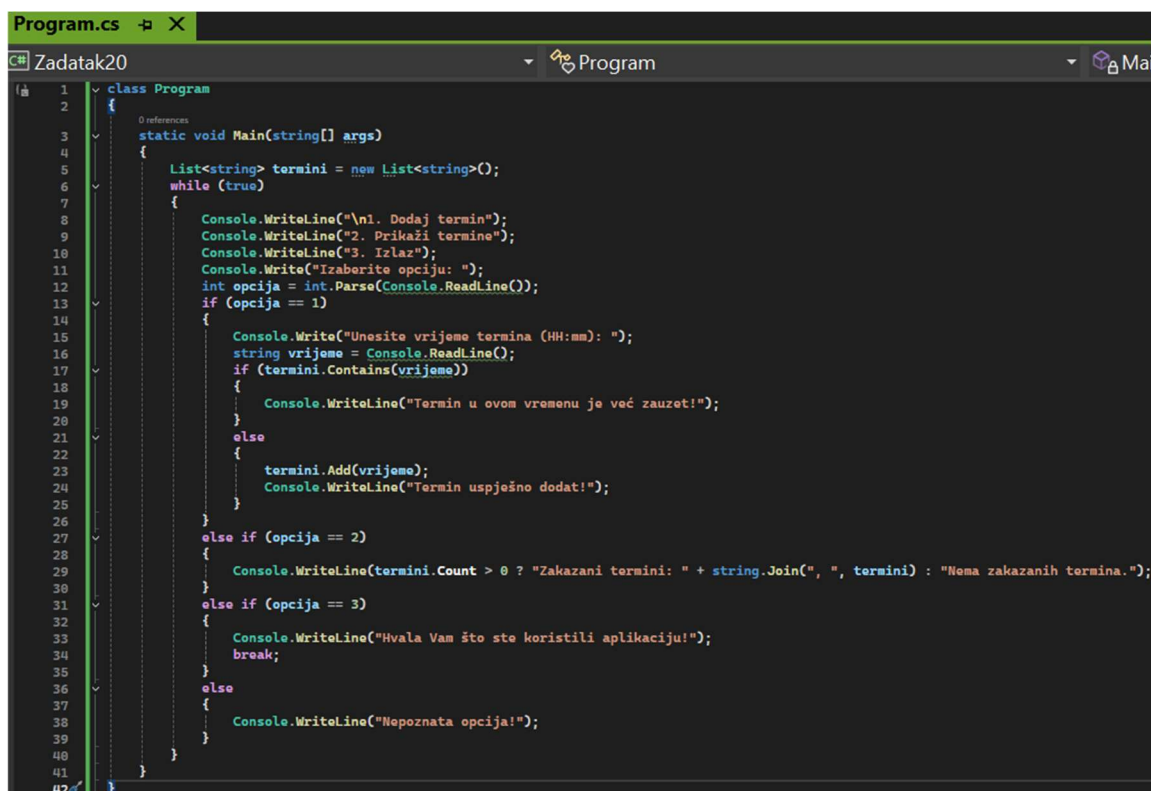
Program omogućava korisnicima da unose podatke o studentima, uključujući njihove ocjene, a zatim izračunava i prikazuje prosječnu ocjene svakog studenta. Atribut ime tipa string čuva ime studenta, a ocjene su svojstva tipa List<int> koje čuva listu ocjena studenata. Metoda ProsjecnaOcjena() izračunava prosječnu ocjenu svakog studenta pojedinačno. List<Student> je lista koja sadrži objekte klase Student, ovdje se čuvaju svi studenti koji će biti unijeti tokom rada programa. For petlja koja ide do brojStudenata omogućava unos podataka svakog studenta i tu se petlja pokreće brojStudenata puta. Tu unosimo ime studenta, broj ocjena koje se konvertuju u int i označavaju koliko student ocjena ima. Kreira se lista ocjena u kojoj će se unositi ocjene, u for petlji koja se ponavlja brojOcjena puta unosimo svaku ocjenu i ocjene se dodaju u listu pomoću ocjena.Add(). Na kraju svakog unosa kreira se novi objekt klase Student sa unijetim imenom i ocjenama, a zatim se dodaje u listu studenti. Potom ide ispis prosječnih ocjena, petlja foreach prolazi kroz sve studente u listi i ispisuje ime studenta i poziva metodu ProsjecnaOcjena() i ispisuje rezultat sa dvije decimale.

21. ZADATAK 20

Napisati program koji omogućava dodavanje i prikaz termina u stomatološkoj ordinaciji. Sistem provjerava da li je termin već zauzet.

21.1. IZRADA ZADATAKA 20

Slika 40. Zadatak 20



```
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         List<string> termini = new List<string>();
6         while (true)
7         {
8             Console.WriteLine("\n1. Dodaj termin");
9             Console.WriteLine("2. Prikazi termine");
10            Console.WriteLine("3. Izlaz");
11            Console.Write("Izaberite opciju: ");
12            int opcija = int.Parse(Console.ReadLine());
13            if (opcija == 1)
14            {
15                Console.Write("Unesite vrijeme termina (HH:mm): ");
16                string vrijeme = Console.ReadLine();
17                if (termini.Contains(vrijeme))
18                {
19                    Console.WriteLine("Termin u ovom vremenu je već zauzet!");
20                }
21                else
22                {
23                    termini.Add(vrijeme);
24                    Console.WriteLine("Termin uspješno dodat!");
25                }
26            }
27            else if (opcija == 2)
28            {
29                Console.WriteLine(termini.Count > 0 ? "Zakazani termini: " + string.Join(", ", termini) : "Nema zakazanih termina.");
30            }
31            else if (opcija == 3)
32            {
33                Console.WriteLine("Hvala Vam što ste koristili aplikaciju!");
34                break;
35            }
36            else
37            {
38                Console.WriteLine("Nepoznata opcija!");
39            }
40        }
41    }
42 }
```

Slika 41. Output zadatka 20

```
Microsoft Visual Studio Debug Console
1. Dodaj termin
2. Prikazi termine
3. Izlaz
Izaberite opciju: 1
Unesite vrijeme termina (HH:mm): 15:00
Termin uspjesno dodat!

1. Dodaj termin
2. Prikazi termine
3. Izlaz
Izaberite opciju: 2
Zakazani termini: 15:00

1. Dodaj termin
2. Prikazi termine
3. Izlaz
Izaberite opciju: 3
Hvala Vam sto ste koristili aplikaciju!
```

21.2. OBRAZLOŽENJE ZADATKA 20

Program omogućava da korisnik upravlja zakazivanjem termina. Koristi se meni od tri opcije: dodavanje termina, prikaz zakazanih termina i izlazak iz programa. List termini će čuvati termine u formatu hh:mm, omogućava dinamičko dodavanje, uklanjanje i pretragu termina. Beskonačna petlja while omogućava korisniku da kontinuirano bira opcije iz menija sve dok ne izabere opciju za izlazak, unos opcije od strane korisnika se konvertuje u int. Korisnik potom treba unijeti vrijeme termina, taj termin se provjerava da li postoji u listi termini, ukoliko postoji ispisuje se da je termin zauzet, ako ne postoji dodaje se u listu pomoću `termini.Add(vrijeme)` i ispisuje se poruka o uspješnosti. Za opciju prikaži termine provjerava se da li lista sadrži elemente, ako je lista prazna ispisuje se da nema zakazanih termina, ukoliko postoji termin ispisuje kad je termin. Za opciju izlaz ispisat će nam zahvalu i prekida se beskonačna petlja i završava se program. Ukoliko korisnik upiše nešto drugo umjesto brojeva 1, 2, ili 3 ispisuje se da je opcija nepoznata.

22. ZAKLJUČAK

U ovom seminarskom radu istraženi su osnovni i napredni koncepti programskog jezika C# kroz implementaciju dvadeset praktičnih zadataka. Svaki zadatak obuhvatio je postavku zadatka, izradu rješenja i obrazloženje korištenih metoda, čime je omogućeno dublje razumijevanje rada sa C#.

Radna hipoteza da se ključni koncepti C# mogu uspješno savladati kroz praktične zadatke potvrđena je. Pored toga, pomoćne hipoteze su pokazale da se kroz rješavanje zadataka unapređuju logičko razmišljanje, algoritamski pristup i sposobnost optimizacije koda.