1. Ponavljanje jednodimenzionalnih nizova

Jednodimenzionalni niz predstavlja skup elemenata istog tipa koji se čuvaju pod zajedničkim nazivom. Svaki element ima svoj položaj, odnosno indeks. Veličina niza određuje se prilikom njegovog kreiranja i kasnije se ne može promeniti.

Deklaracija i inicijalizacija niza

Deklaracijom se određuju tip elemenata i naziv niza:

```
int[] brojevi;
```

Niz možemo naknadno kreirati i odrediti mu broj elemenata:

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

Ovako je napravljen niz od pet celih brojeva. Početna vrednost svakog elementa je `0`.

Deklaracija, kreiranje i dodeljivanje početnih vrednosti mogu se napisati zajedno:

```
int[] brojevi = { 12, 7, 5, 18, 9 };
```

Moguće je koristiti i duži oblik:

```
int[] brojevi = new int[] { 12, 7, 5, 18, 9 };
```

Svi elementi jednog niza moraju biti istog tipa.

```
string[] gradovi = { "Subotica", "Novi Sad", "Sombor" };
double[] temperature = { 18.5, 20.2, 17.8 };
```

Indeksiranje elemenata

Elementima niza pristupa se pomoću indeksa zapisanog između uglastih zagrada. Indeksiranje počinje od nule.

```
int[] brojevi = { 12, 7, 5, 18, 9 };

Console.WriteLine(brojevi[0]); // 12
Console.WriteLine(brojevi[2]); // 5
Console.WriteLine(brojevi[4]); // 9
```

Raspored elemenata može se predstaviti ovako:

Indeks	**0**	**1**	**2**	**3**	**4**
Vrednost	12	7	5	18	9

Vrednost određenog elementa može se promeniti:

```
brojevi[1] = 20;
```

Nakon ove naredbe niz sadrži: 12 20 5 18 9

Svojstvo Length

Svojstvo Length vraća ukupan broj elemenata niza.

```
int[] brojevi = { 12, 7, 5, 18, 9 };
Console.WriteLine(brojevi.Length); // 5
```

Pošto indeksiranje počinje od nule, indeks poslednjeg elementa je za jedan manji od dužine niza:

```
int poslednjiIndeks = brojevi.Length - 1;

Console.WriteLine(brojevi[poslednjiIndeks]); // 9
```

Dakle:

broj elemenata: Length

indeks poslednjeg: Length - 1

Unos elemenata pomoću petlje

Kada korisnik treba da unese vrednosti, prvo se kreira niz odgovarajuće dužine.

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

for (int i = 0; i < brojevi.Length; i++)
{
    Console.Write($"Unesite element sa indeksom {i}: ");
    brojevi[i] = int.Parse(Console.ReadLine());
}
```

Promenljiva i predstavlja indeks elementa koji se trenutno unosi.

Ispis elemenata pomoću petlje

Elementi se mogu ispisati pomoću iste vrste petlje:

```
for (int i = 0; i < brojevi.Length; i++)
{
    Console.WriteLine($"brojevi[{i}] = {brojevi[i]}");
}
```

Ako niz sadrži vrednosti 4, 8, 3, 6, 2, rezultat je:

```
brojevi[0] = 4
brojevi[1] = 8
brojevi[2] = 3
brojevi[3] = 6
brojevi[4] = 2
```

Za ispis svih elemenata u istom redu može se koristiti:

```
for (int i = 0; i < brojevi.Length; i++)
{
    Console.Write(brojevi[i] + " ");
}
```

Prolazak pomoću for i foreach

Petlja for koristi se kada nam je potreban indeks elementa.

```
for (int i = 0; i < brojevi.Length; i++)
{
    Console.WriteLine($"Indeks: {i}, vrednost: {brojevi[i]}");
}
```

Petlja foreach prolazi direktno kroz vrednosti elemenata:

```
foreach (int broj in brojevi)
{
    Console.WriteLine(broj);
}
```

Osnovna razlika:

for	foreach
Omogućava pristup indeksu	Pristupa direktno vrednosti
Može menjati element niza preko indeksa	Najčešće se koristi za čitanje
Pogodan kada je položaj važan	Jednostavniji za prolazak kroz ceo niz

2. Uvod u dvodimenzionalne nizove

Dvodimenzionalni niz koristi se za čuvanje podataka raspoređenih u redove i kolone. Zbog takvog rasporeda može se posmatrati kao tabela.

Primeri podataka koji se mogu predstaviti dvodimenzionalnim nizom su poeni učenika na više zadataka, temperature izmerene tokom više dana, raspored sedišta, polja na tabli za igru, elementi matematičke matrice.

Dvodimenzionalni niz kao tabela

Posmatrajmo sledeću tabelu:

	Kolona 0	**Kolona 1**	**Kolona 2**
Red 0	4	7	2
Red 1	9	1	5

Ona ima dva reda; tri kolone; ukupno šest elemenata.

U C# jeziku može se predstaviti dvodimenzionalnim nizom:

```
int[,] brojevi =
{
    { 4, 7, 2 },
    { 9, 1, 5 }
};
```

Svaki unutrašnji par vitičastih zagrada predstavlja jedan red:

```
{ 4, 7, 2 } // prvi red
{ 9, 1, 5 } // drugi red
```

Redovi i kolone

Elementi dvodimenzionalnog niza raspoređeni su u redove i kolone.

U primeru:

```
int[,] brojevi =
{
    { 4, 7, 2 },
    { 9, 1, 5 }
};
```

prvi red sadrži: 4 7 2

Drugi red sadrži: 9 1 5

Prva kolona sadrži brojeve 4 i 9, druga kolona brojeve 7 i 1, a treća brojeve 2 i 5.

Dva indeksa

Za pristup elementu dvodimenzionalnog niza potrebna su dva indeksa:

`brojevi[indeksReda, indeksKolone]`

prvi indeks označava red, a drugi indeks označava kolonu.

Na primer: `brojevi[0, 2]` označava element koji se nalazi u redu sa indeksom 0 i koloni sa indeksom 2.

Njegova vrednost za niz iz primera je 2.

Položaji elemenata mogu se prikazati ovako:

	Kolona 0	**Kolona 1**	**Kolona 2**
Red 0	brojevi[0,0]	brojevi[0,1]	brojevi[0,2]
Red 1	brojevi[1,0]	brojevi[1,1]	brojevi[1,2]

Sa konkretnim vrednostima:

	Kolona 0	**Kolona 1**	**Kolona 2**
Red 0	[0,0] = 4	[0,1] = 7	[0,2] = 2
Red 1	[1,0] = 9	[1,1] = 1	[1,2] = 5

Indeksiranje počinje od nule

Kao i kod jednodimenzionalnih nizova, indeksiranje počinje od nule.

Niz koji ima dva reda koristi indekse redova 0 i 1

Niz koji ima tri kolone koristi indekse kolona 0, 1 i 2

Zato je poslednji postojeći element u nizu dimenzija 2×3: `brojevi[1, 2]`

Deklaracija dvodimenzionalnog niza

Prilikom deklaracije između uglastih zagrada navodi se zarez: `int[,] brojevi;`

Ovim je deklarisana promenljiva koja može sadržati dvodimenzionalni niz celih brojeva.

Niz se zatim može kreirati navođenjem broja redova i kolona: `brojevi = new int[2, 3];`

Broj 2 predstavlja broj redova, a broj 3 broj kolona.

Deklaracija i kreiranje mogu se napisati zajedno: `int[,] brojevi = new int[2, 3];`

Svi elementi ovako kreiranog niza imaju početnu vrednost 0.

Inicijalizacija dvodimenzionalnog niza

Vrednosti elemenata mogu se navesti prilikom kreiranja niza:

```
int[,] brojevi =
{
    { 4, 7, 2 },
    { 9, 1, 5 }
};
```

Može se koristiti i duži zapis:

```
int[,] brojevi = new int[,]
{
    { 4, 7, 2 },
    { 9, 1, 5 }
};
```

Svi redovi pravougaonog dvodimenzionalnog niza moraju imati isti broj elemenata.

Ovakva inicijalizacija nije ispravna:

```
int[,] brojevi =
{
    { 4, 7, 2 },
    { 9, 1 }
};
```

Prvi red ima tri, a drugi samo dva elementa.

Pristup pojedinačnom elementu

Pojedinačni elementi mogu se pročitati navođenjem njihovih indeksa:

```
int[,] brojevi =
{
    { 4, 7, 2 },
    { 9, 1, 5 }
};

Console.WriteLine(brojevi[0, 0]); // 4
Console.WriteLine(brojevi[0, 2]); // 2
Console.WriteLine(brojevi[1, 1]); // 1
```

Vrednost elementa može se i promeniti: `brojevi[1, 1] = 10;`

Pre promene drugi red je bio: 9 1 5

Nakon promene postaje: 9 10 5

Svojstvo Length

Svojstvo Length vraća ukupan broj elemenata u nizu.

```
int[,] brojevi = new int[2, 3];

Console.WriteLine(brojevi.Length);
```

Rezultat je: 6

Niz ima dva reda i tri kolone, pa je ukupan broj elemenata: $2 \cdot 3 = 6$

Length ne govori posebno koliko niz ima redova, a koliko kolona.

Metoda GetLength()

Metoda GetLength() vraća broj elemenata u određenoj dimenziji niza.

```
int brojRedova = brojevi.GetLength(0);
int brojKolona = brojevi.GetLength(1);

Console.WriteLine($"Broj redova: {brojRedova}");
Console.WriteLine($"Broj kolona: {brojKolona}");
```

Rezultat:

```
Broj redova: 2
Broj kolona: 3
```

Broj prosleđen metodi označava dimenziju:

```
brojevi.GetLength(0) // prva dimenzija – redovi
brojevi.GetLength(1) // druga dimenzija – kolone
```

Razlika između Length i GetLength()

Za niz `int[,] tabela = new int[3, 4];` važi:

Izraz	Rezultat	Značenje
tabela.Length	12	Ukupan broj elemenata
tabela.GetLength(0)	3	Broj redova
tabela.GetLength(1)	4	Broj kolona

Indeks poslednjeg reda je `tabela.GetLength(0) - 1`

Indeks poslednje kolone je `tabela.GetLength(1) - 1`

Poslednjem elementu niza može se pristupiti ovako:

```
int poslednjiRed = tabela.GetLength(0) - 1;
int poslednjaKolona = tabela.GetLength(1) - 1;

Console.WriteLine(tabela[poslednjiRed, poslednjaKolona]);
```

3. Prolazak kroz dvodimenzionalni niz

Za prolazak kroz dvodimenzionalni niz najčešće se koriste dve ugnežđene for petlje, spoljašnja petlja bira red, a unutrašnja petlja prolazi kroz sve kolone izabranog reda.

Posmatrajmo niz:

```
int[,] brojevi =
{
    { 4, 7, 2 },
    { 9, 1, 5 }
};
```

Spoljašnja petlja – prolazak kroz redove

Broj redova dobija se pomoću: `brojevi.GetLength(0)`

Zato spoljašnja petlja izgleda ovako:

```
for (int red = 0; red < brojevi.GetLength(0); red++)
{
    // Obrada jednog reda
}
```

Promenljiva red redom dobija vrednosti 0, 1.

Za svaku vrednost promenljive red obrađuje se jedan red tabele.

Unutrašnja petlja – prolazak kroz kolone

Broj kolona dobija se pomoću: `brojevi.GetLength(1)`

Unutrašnja petlja izgleda ovako:

```
for (int kolona = 0; kolona < brojevi.GetLength(1); kolona++)
{
    // Obrada jednog elementa izabranog reda
}
```

Promenljiva kolona redom dobija vrednosti 0, 1, 2.

Ugnežđene petlje

Kada se petlje spoje, dobija se prolazak kroz svaki element dvodimenzionalnog niza:

```
for (int red = 0; red < brojevi.GetLength(0); red++)
{
    for (int kolona = 0; kolona < brojevi.GetLength(1); kolona++)
    {
        Console.WriteLine(brojevi[red, kolona]);
    }
}
```

Unutrašnja petlja mora se završiti za svaki red. Redosled posećivanja elemenata je:

```
brojevi[0,0]
brojevi[0,1]
brojevi[0,2]
brojevi[1,0]
brojevi[1,1]
brojevi[1,2]
```

Odgovarajuće vrednosti su 4, 7, 2, 9, 1, 5

Opšti obrazac prolaska

Za bilo koji dvodimenzionalni niz može se koristiti sledeći obrazac:

```
for (int red = 0; red < niz.GetLength(0); red++)
{
    for (int kolona = 0; kolona < niz.GetLength(1); kolona++)
    {
        // Obrada elementa niz[red, kolona]
    }
}
```

Nazivi promenljivih red i kolona jasno pokazuju šta predstavljaju. Umesto njih često se koriste kraći nazivi i i j:

```
for (int i = 0; i < niz.GetLength(0); i++)
{
    for (int j = 0; j < niz.GetLength(1); j++)
    {
        Console.WriteLine(niz[i, j]);
    }
}
```

U tom slučaju i predstavlja indeks reda, a j predstavlja indeks kolone.

Ispis elemenata u obliku tabele

Da bi se sačuvao oblik tabele, elementi jednog reda ispisuju se pomoću Console.Write(). Prelazak u novi red izvršava se tek nakon završetka unutrašnje petlje.

```
for (int red = 0; red < brojevi.GetLength(0); red++)
{
    for (int kolona = 0; kolona < brojevi.GetLength(1); kolona++)
    {
        Console.Write(brojevi[red, kolona] + "\t");
    }
    Console.WriteLine();
}
```

Rezultat:

```
4   7   2
9   1   5
```

Naredba `Console.Write(brojevi[red, kolona] + "\t");` ispisuje element i ostaje u istom redu.

Naredba `Console.WriteLine();` nalazi se posle unutrašnje petlje jer se novi red započinje tek kada su ispisane sve kolone trenutnog reda.

Unos elemenata sa tastature

Za unos se prvo kreira niz određene veličine: `int[,] brojevi = new int[2, 3];`

Zatim se pomoću dve petlje dolazi do svakog elementa:

```
for (int red = 0; red < brojevi.GetLength(0); red++)
{
    for (int kolona = 0; kolona < brojevi.GetLength(1); kolona++)
    {
        Console.Write($"Unesite element [{red},{kolona}]: ");
        brojevi[red, kolona] = int.Parse(Console.ReadLine());
    }
}
```

Poruka korisniku prikazuje mesto elementa koji se trenutno unosi:

```
Unesite element [0,0]:
Unesite element [0,1]:
Unesite element [0,2]:
Unesite element [1,0]:
Unesite element [1,1]:
Unesite element [1,2]:
```

Nakon unosa, niz se može prikazati pomoću drugog para petlji:

```
Console.WriteLine("Niz koji ste uneli:");

for (int red = 0; red < brojevi.GetLength(0); red++)
{
    for (int kolona = 0; kolona < brojevi.GetLength(1); kolona++)
    {
        Console.Write(brojevi[red, kolona] + "\t");
    }

    Console.WriteLine();
}
```

4. Niz nizova

Nepravilni niz, odnosno niz nizova (*jagged array*), jeste niz čiji je svaki element drugi niz.

Za razliku od pravougaonog dvodimenzionalnog niza, unutrašnji nizovi ne moraju imati isti broj elemenata. Zbog toga svaki „red" može imati drugačiju dužinu.

Niz čiji su elementi drugi nizovi

Niz nizova celih brojeva deklariše se ovako: `int[][] brojevi;`

Prve uglaste zagrade predstavljaju spoljašnji niz, dok druge označavaju da su njegovi elementi takođe nizovi.

Možemo prvo kreirati spoljašnji niz: `int[][] brojevi = new int[3][];`

Ovim je napravljen niz koji može sadržati tri unutrašnja niza. Unutrašnji nizovi još nisu kreirani.

Njih je potrebno posebno napraviti:

```
brojevi[0] = new int[4];
brojevi[1] = new int[2];
brojevi[2] = new int[5];
```

Dobijena struktura ima četiri elementa u prvom redu, dva elementa u drugom redu i pet elemenata u trećem redu.

Može se predstaviti ovako:

Red 0: 0 0 0 0

Red 1: 0 0

Red 2: 0 0 0 0 0

Svaki unutrašnji niz predstavlja jedan red, ali redovi nisu obavezno iste dužine.

Inicijalizacija niza nizova

Vrednosti se mogu navesti prilikom kreiranja:

```
int[][] brojevi =
{
    new int[] { 4, 7, 2, 9 },
    new int[] { 5, 1 },
    new int[] { 8, 3, 6 }
};
```

Struktura izgleda ovako:

Red 0: 4 7 2 9

Red 1: 5 1

Red 2: 8 3 6

Svaki unutrašnji niz ima sopstveni par vitičastih zagrada i može imati različit broj elemenata.

Moguće je koristiti i kraći oblik:

```
int[][] brojevi =
{
    new[] { 4, 7, 2, 9 },
    new[] { 5, 1 },
    new[] { 8, 3, 6 }
};
```

C# na osnovu navedenih vrednosti zaključuje da su unutrašnji nizovi tipa int[].

Pristup elementima

Kod pravougaonog dvodimenzionalnog niza dva indeksa pišu se unutar jednog para uglastih zagrada: `tabela[red, kolona]`

Kod niza nizova svaki indeks piše se u posebnom paru zagrada: `brojevi[red][kolona]`

Na primeru niza brojevi kreiranog u podnaslovu iznad:

```
Console.WriteLine(brojevi[0][2]); // 2
Console.WriteLine(brojevi[1][0]); // 5
Console.WriteLine(brojevi[2][1]); // 3
```

Izraz `brojevi[2]` predstavlja ceo unutrašnji niz: 8 3 6

Izraz `brojevi[2][1]` predstavlja element sa indeksom 1 unutar tog niza. Njegova vrednost je 3.

Vrednost elementa može se promeniti `brojevi[1][0] = 10;` Drugi red tada postaje: 10 1

Dužina spoljašnjeg i unutrašnjih nizova

Svojstvo Length spoljašnjeg niza vraća broj unutrašnjih nizova, odnosno broj redova: `Console.WriteLine(brojevi.Length);` Rezultat: 3

Dužina pojedinačnog reda dobija se pomoću `brojevi[indeksReda].Length`

Na primeru gore kreiranog niza:

```
Console.WriteLine(brojevi[0].Length); // 4
Console.WriteLine(brojevi[1].Length); // 2
Console.WriteLine(brojevi[2].Length); // 3
```

Svaki red ima sopstveno svojstvo Length.

Kod niza nizova ne koristi se `brojevi.GetLength(0)` i `brojevi.GetLength(1)`.

Spoljašnji niz i svaki unutrašnji niz posmatraju se kao posebni jednodimenzionalni nizovi.

Prolazak kroz niz nizova

Za prolazak se koriste dve ugnežđene petlje. Spoljašnja petlja prolazi kroz unutrašnje nizove, a unutrašnja kroz elemente konkretnog niza.

```
for (int red = 0; red < brojevi.Length; red++)
{
    for (int kolona = 0; kolona < brojevi[red].Length; kolona++)
    {
        Console.Write(brojevi[red][kolona] + " ");
    }

    Console.WriteLine();
}
```

Rezultat:

4 7 2 9

5 1

8 3 6

Granica unutrašnje petlje zavisi od trenutnog reda `kolona < brojevi[red].Length`

Ne može se koristiti jedna zajednička dužina kolona jer redovi mogu imati različit broj elemenata.

Kreiranje redova različite dužine

Dužina svakog unutrašnjeg niza može se uneti sa tastature:

```
Console.Write("Unesite broj redova: ");
int brojRedova = int.Parse(Console.ReadLine());

int[][] brojevi = new int[brojRedova][];

for (int red = 0; red < brojevi.Length; red++)
{
    Console.Write($"Broj elemenata u redu {red}: ");
    int brojElemenata = int.Parse(Console.ReadLine());

    brojevi[red] = new int[brojElemenata];
}
```

Nakon kreiranja redova mogu se uneti njihovi elementi:

```
for (int red = 0; red < brojevi.Length; red++)
{
    for (int kolona = 0; kolona < brojevi[red].Length; kolona++)
    {
        Console.Write($"Element [{red}][{kolona}]: ");
        brojevi[red][kolona] = int.Parse(Console.ReadLine());
    }
}
```

Svaki red se najpre posebno kreira, a tek zatim se može pristupiti njegovim elementima.

Važnost kreiranja unutrašnjih nizova

Nakon naredbe `int[][] brojevi = new int[3][];` napravljen je samo spoljašnji niz. Unutrašnji nizovi još ne postoje.

Zato sledeća naredba izaziva grešku `brojevi[0][0] = 5;`

Pre pristupa elementu mora se kreirati prvi unutrašnji niz:

```
brojevi[0] = new int[4];
brojevi[0][0] = 5;
```

Ako unutrašnji niz nije kreiran, njegova vrednost je null, pa pristup njegovom elementu izaziva grešku NullReferenceException.

Razlika između int[,] i int[][]

Osobina	int[,]	int[][]
Naziv	Dvodimenzionalni niz	Niz nizova
Oblik	Pravougaona tabela	Redovi različitih dužina
Deklaracija	int[,] tabela	int[][] nizovi
Kreiranje	new int[3, 4]	new int[3][]
Pristup elementu	tabela[red, kolona]	nizovi[red][kolona]
Broj redova	GetLength(0)	Length
Broj kolona	GetLength(1)	nizovi[red].Length

Osobina	int[,]	int[][]
Unutrašnji redovi	Deo jedne strukture	Posebni nizovi
Dužina redova	Uvek jednaka	Može biti različita