Programski jezik C#

Znaki, nizi in tabele

Matija Lokar in Srečo Uranič

V 0.83

november 2008

Predgovor

Omenjeno gradivo predstavlja četrti del gradiv, namenjenih predmetu Programiranje 1 na višješolskem študiju Informatika. V predhodnih delih smo spoznali osnovne podatkovne tipe (int, double, bool), prireditveni stavek, pogojni stavek, zanke ter osnovne prijeme za branje/izpisovanje podatkov preko konzole.

Pokriva znake (tip char), nize (string) in tabele v jeziku C#. Glede samega besedila velja tisto, kar je bilo napisano v predgovorih prejšnjih delov gradiv, torej – ne poveva vsega, določene stvari poenostaviva, veliko je zgledov ... .

Gradivo vsekakor ni dokončano in predstavlja delovno različico. V njem so zagotovo napake (upava, da čimmanj), za katere se vnaprej opravičujeva. Da bo lažje spremljati spremembe, obstaja razdelek Zgodovina sprememb, kamor bova vpisovala spremembe med eno in drugo različico. Tako bo nekomu, ki si je prenesel starejšo različico, lažje ugotoviti, kaj je bilo v novi različici spremenjeno.

Matija Lokar in Srečo Uranič

Kranj, oktober 2008

Zgodovina sprememb

8. 10. 2008: Različica V0.8 – prva, ki je na voljo javno
31. 10. 2008: Različica V0.81 – oblikovanje in manjši popravki, dodana zanka foreach, dodane dvodimenzionalne tabele.
10. 11. 2008: Različica V0.82 – manjši tipkarski popravki, par besed v predgovoru.
16.11.2008: Različica V0.83 – manjši tipkarski popravki

KAZALO

Znaki (tip char)

V spremenljivki lahko hranimo tudi en znak. V ta namen uporabljamo podatkovni tip **char** (ang. character). To so črke, številke, presledek, ločila, ... Znake pišemo med enojnimi narekovaji.

Primer:

```
char znak = 'n';
char oznakaVrat = 'N';
char zacetnica = 'M';
```

Primer uporabe:

```
static void Main(string[] args)
{
    char znak_a = 'a';
    Console.WriteLine(znak_a);
} // main
```

Program prevedemo in poženemo:

```
a
```

C# obravnava znake kot "majhna" števila. Zato smo v spremenljivko *znak_a* pravzaprav shranili kodo znaka mali a. Ta koda je neko naravno število. Na vsako spremenljivko tipa *char* lahko gledamo bodisi kot na znak, bodisi kot na število. Ker na znake lahko gledamo tudi kot na števila, smemo napisati tudi tak program.

```
static void Main(string[] args)
{
     char znak_a = 'a';

     Console.WriteLine(znak_a + znak_a);
     Console.ReadKey();
} // main
```

Program prevedemo in poženemo:

```
194
```

V spremenljivko tipa *char* pravzaprav shranimo kodo znaka. Zato smo v spremenljivko *znak_a* shranili kodo znaka mali a, ki je neko naravno število (v našem primeru *97*). In ker je med dvema "številoma" operator *+*, se izvede operacija seštevanja, kot smo že navajeni. Torej se seštejeta dve številski vrednosti malega znaka a in vsota se izpiše.

Razlikovati moramo med znakom in nizom, ki vsebuje en znak. Tako znak 'm' ni enak nizu, ki vsebuje le znak m, torej "m".

```
string znakKotNiz = "m";
char znakKotZnak = 'm';
```

Abeceda

Napišimo program, ki bo izpisal angleško abecedo. Prvič naprej, drugič pa v obratnem vrstnem redu. Izrabili bomo dejstvo, da so znaki "števila". Ker z njimi lahko računamo, to pomeni tudi, da jih lahko uporabimo kot števce v zankah.

```
1:    static void Main(string[] args)
2:    {
3:          for(char i = 'a'; i <= 'z'; i++)// abeceda naprej
4:          {
5:                Console.Write (i + " ");
6:          } // for
7:          Console.WriteLine ("\n"); // vmesna prazna vrsta
8:          for(char j = 'z'; j >= 'a'; j--)  // abeceda nazaj
9:          {
10:               Console.Write (j + " ");
11:         }  // for
12:    }
```

Program prevedemo in poženemo:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
z y x w v u t s r q p o n m l k j i h g f e d c b a
```

Opis programa.
Kot smo že rekli, lahko na znake gledamo kot na števila. Zato smo v spremenljivki *i* pravzaprav shranili kodo znaka mali *a*. Ta koda je neko naravno število. Če so znaki kodirani po ASCII[1] standardu, se v *a* shrani število *97*. Preveri se pogoj in ker je *97 <= 122* (koda znaka *'z'*) se izpiše znak s kodo *97*, torej *a*. Vrednost v spremenljivki *i* se poveča za ena (*i++*). Zopet se preveri pogoj in ker je resničen, se izpiše znak s kodo *98*. Na ta način se zanka *for* izvaja, dokler je vrednost v spremenljivki *i* manjša ali enaka *'z'* (*122*). Ko je v spremenljivki *i* vrednost *122*, je pogoj izpolnjen in 5. vrstica izpiše znak s kodo 122, torej *z*. Sedaj spremenljivka *i* dobi vrednost *123*. Preveri se pogoj. Ker *123* ni manjše ali enako *122*, pogoj ni več izpolnjen. Zato se zanka konča in program se nadaljuje v vrstici 7. Izpiše se prazna vrsta. Program se nadaljuje v vrstici 8. Če želimo izpisati abecedo v obratnem vrstnem redu, začnemo izpisovati znake od zadnjega proti prvemu. V spremenljivki j shranimo kodo znaka mali *z* (*122*). Preveri se pogoj in ker je *122 >= 97* (koda znaka mali *a*), se izpiše znak s kodo *122*, torej *z*. Vrednost v spremenljivki *j* se zmanjša za ena (*j--*) in zopet se preveri pogoj. Ker je resničen, se izpiše znak s kodo *121*. Na ta način se zanka *for* izvaja, dokler je vrednost v spremenljivki *j* večja ali enaka *'a'* (*97*). Ko je v spremenljivki vrednost *97*, je pogoj izpolnjen in 10. vrstica izpiše znak s kodo *97*, torej *a*. Sedaj spremenljivka *i* dobi vrednost *96*. Preveri se pogoj. Ker *96* ni večje ali enako *97*, pogoj ni več izpolnjen, zato se zanka konča.

Primerjanje znakov

Ker v spremenljivko tipa *char* shranimo kodo znaka, znake primerjamo tako kot števila z operatorjem enakosti (==). S tem se v bistvu primerja koda znaka. Ker ima vsak znak svojo kodo, lahko dobimo vrednost *true* samo, kadar primerjamo enaka znaka, drugače dobimo *false*. Poglejmo si primer.

```
static void Main(string[] args)
{
     char znak_a = 'a';
     char znak_A = 'A';
     Console.WriteLine(znak_a == znak_A);
```

1 V računalništvu oznaka za splošno uveljavljeno 8-bitno kodo za kodiranje običajnih črk in znakov, s katerimi se zapisuje besedila.

```
    Console.WriteLine(znak_a == 'a');
} // main
```

Program prevedemo in poženemo:

```
False
True
```

Kot smo že povedali, C# primerja znake po njihovih celoštevilčnih kodah. Ker koda malega in velikega znaka a ni enaka, se izpiše *false*. V naslednji vrstici primerjamo vrednost spremenljivke *znak_a* in znakovne *konstante 'a'*. Ker je njuna celoštevilčna koda enaka, se izpiše *true*.

Za primerjanje znakov je pomembno vedeti, kako so znaki zloženi v kodnih tabelah. Velja:

- 'a' < 'b' < 'c' < … < 'z'
- '0' < '1' < … < '9'
- 'A' < 'B' < 'C' < … < 'Z'

Vmes v teh treh zaporedjih ni nobenih drugih znakov. Če torej napišemo pogoj

```
('0' <= preverjaniZnak) && (preverjaniZnak <= '9')
```

bo ta imel vrednost *true* (bo torej izpolnjen), če je v spremenljivki preverjaniZnak števka. Podobno z

```
('a' <= malaČrka) && (malaČrka <= 'z')
```

preverjamo, če je v spremenljivki malaČrka res mala črka. Pogoj pomeni, da je znak v tej spremenljivki "desno" od 'a' in "levo" od 'z'. In ker med 'a' in 'z' ni drugih znakov, temu pogoj ustrezajo le male črke.

Pretvarjanje znakov

S pomočjo operatorja (char) celo število lahko pretvorimo v znak. Tako na primer velja

`(char)('a')` → 'a'

In ker z znaki lahko računamo in so, kot smo videli v zgornjem razdelku, črke lepo "zložene skupaj", je seveda res tudi, da

`(char)('a' + 2)` → 'c'

To lahko izrabimo, če moramo slučajno malo črko pretvoriti v veliko ali obratno. Kratek premislek pokaže, da velja:

`(char)('A' + 'k' - 'a')` → 'K'
`(char)('a' + 'M' - 'A')` → 'm'

Kako to? Če želimo malo črko 'k' pretvoriti v veliko, vemo, da bo razlika v kodi med 'k' in 'a' enaka razliki kod 'K' in 'A'. Torej moramo od kode 'A' iti naprej točno za razliko med kodo 'k' in 'a', da pridemo do kode znaka 'K'. Če je v spremenljivki maliZnak torej neka mala črka, bomo z

```
(char)('A' + maliZnak - 'a')
```

dobili ustrezno veliko črko.

Oglejmo si še zgled uporabe.

Geslo

Iz oddelka za varovanje podatkov smo dobili nalogo, da napišemo program za samodejno generiranje gesel. Sodelavci tega oddelka želijo gesla naslednje oblike:

- najprej dve naključni števki,
- potem tri naključne velike črke,
- nato še ena naključna števka in
- na koncu naključno trimestno število.

Pri generiranju naključnih črk si bomo pomagali z dejstvom, da je znak predstavljen kar s številom, ki predstavlja njegovo kodo. Z njim torej lahko računamo. Izraz geslo.Next('A', 'Z' + 1) je torej koda neke velike črke. Za pretvarjanje iz celega števila v znak bomo uporabili operator (char).

```
C2:  static void Main(string[] args)
C3:  {    // Ustvarimo generator naključnih števil
C4:      Random geslo = new Random();
C5:      // Določimo dve števki
C6:      int stev1 = geslo.Next(10);
C7:      int stev2 = geslo.Next(10);
C8:      // Določimo tri naključne črke (velike tiskane)
C9:      char c1 = (char)geslo.Next('A', 'Z' + 1);
C10:     char c2 = (char)geslo.Next('A', 'Z' + 1);
C11:     char c3 = (char)geslo.Next('A', 'Z' + 1);
C12:     // Določimo eno naključno števko
C13:     int stev3 = geslo.Next(10);
C14:
C15:     // Določimo naključno tromestno število
C16:     int s = geslo.Next(100, 1000);
C17:     // Izpis gesla
C18:     Console.WriteLine("Geslo: "+stev1+stev2 +c1 +c2 +c3+stev3+ s);
C19:     Console.ReadKey();
C20: }
C21:
C22:
```

Zapis na zaslonu:

```
Geslo: 99OSF8654
```

Razlaga

Najprej ustvarimo generator naključnih števil. Nato generiramo posamezne elemente gesla (C7 - C18). Za generiranje naključne črke uporabimo izraz geslo.Next('A', 'Z' + 1). Z omenjenim izrazom določimo kodo črke. Ker se koda črke (tipa int) ne spremeni v črko (tip char) samodejno, pred omenjenim izrazom zapišemo operator (char). S tem operatorjem iz kode dobimo naključno črko, ki jo nato priredimo spremenljivki. Po generiranju združimo vse elemente in jih izpišemo na zaslon (C20).

Naloge (tip char)

Napišite program, ki bere posamezne znake in jih kodira v števila na naslednji način:

- Ne loči velike in male črke,
- črke od a do e (A do E) kodira po vrsti s števili od 1 do 5,
- številke 0 do 9 kodira po vrsti s števili od 6 do 15,
- črke od f do z (F do Z) kodira s števili od 16 do 36 v obratnem vrstnem redu,
- vse ostale znake zakodira s številom 0.

Program naj za vsak posamezen znak izpiše na zaslon znak = koda. Branje se konča, če je prebran znak *.
Namig: Za pretvarjanje iz tipa string v tip char si pomagajte z metodo char.Parse(niz).

Primer kodiranja:

Vnesi znak: a
a = 1

Vnesi znak: Z
Z = 16

Nizi (tip string)

Pogosteje kot posamezne znake uporabljamo nize znakov (ang. string). Za delo z nizi je v jeziku C# definiran tip *string*. Željen niz navedemo med dvojnimi narekovaji. Podatkovni tip string (niz) torej označuje zaporedje znakov. Spremenljivke tega tipa v C# lahko inicializiramo takole:

```
string var5;
var5 = "Lokomotiva";
string niz = "Pozdravljeni!";
string presledek = " ";
```

Dva niza staknemo (združimo) z operatorjem za združevanje (+). To smo že večkrat srečali pri izpisovanju.

```
static void Main(string[] args)
{
     string niz1 = "Dobro ";
     string niz2 = "jutro.";
     string niz3 = niz1 + niz2;

     Console.WriteLine(niz3);
     Console.ReadKey();
}
```

Program prevedemo in poženemo:

```
Dobro jutro.
```

Niz je lahko sestavljen iz enega ali več znakov, poznamo pa tudi t.i. prazen niz, ki ne vsebuje nobenega znaka.

```
string prazenNiz = "";
```

Dostop do znakov v nizu

Niz je lahko sestavljen iz enega ali več znakov, poznamo pa tudi t.i. prazen niz, ki ne vsebuje nobenega znaka. Vsak znak, ki je vsebovan v nizu, ima svoj indeks. Do posameznega znaka v nizu dostopamo s pomočjo oglatih oklepajev ([]), kjer navedemo indeks znaka. Z `niz[i]`
torej dobimo znak z indeksom i iz niza niz.

- `"Blabla"[3]` → `b`
- `string ime = "Matija";`
- `ime[1]` → `a`

Indeksiranje znakov se prične z 0 in se konča z dolžina niza - 1.

Primer:

Niz	"D	o	b	e	r		d	a	n	."
Indeks	0	1	2	3	4	5	6	7	8	9

Z uporabo oglatih oklepajev ne moremo spreminjati znakov v nizu, temveč jih lahko le beremo. Izraz niz[indeks] se torej ne sme pojaviti na levi strani prireditvenega stavka.

Dolžina niza

Da zvemo dolžino niza, uporabljamo lastnost Length.

Primer:

```
string n = "Rock Otocec";
int dolzina = n.Length;
```

Vrednost spremenljivke dolzina je 11.

- `string priimek = "Lokar";`
- `priimek.Length` → 5
- `"matija".Length` → 6
- `(priimek + " " + "bla").Length` → 9

Zgledi

Obratni izpis

Napišimo program, ki bo vneseni niz izpisal v obratnem vrstnem redu. Torej bo npr. niz "Matija" izpisal kot ajitaM.

Tu si bomo pomagali z *[i]*, ki vrne znak z indeksom *i* ter z lastnostjo *Length*, ki vrne dolžino niza. Ker v nizih znake indeksiramo od 0 dalje, nam *niz.[i]* vrne i+1-ti znak niza *niz*.

Ideja programa je v tem, da naredimo zanko, v kateri izpišemo posamezen znak. Začnemo pri zadnjem znaku in zmanjšujemo števec, ki pomeni indeks, dokler ne pridemo do vrednosti števca 0 (torej do zadnjega znaka), ko še zadnjič izvedemo zanko.

```
1:     static void Main(string[] args)
2:     {
3:         Console.WriteLine("Vnesi niz:");
4:         string beri = Console.ReadLine();
5:         Console.WriteLine("Vnešeni niz je: " + beri);
6:
7:         for(int i = beri.Length - 1; i >= 0; i--)
8:         {
9:             Console.Write (beri[i]);
10:        } // for
11:        Console.WriteLine ("\n");
12:    } // main
```

Opis programa
Če želimo niz izpisati v obratnem vrstnem redu, bomo izpisovali znake od zadnjega proti prvemu. Recimo, da smo vnesli niz *"abeceda"*. Spremenljivki *i* se najprej priredi začetna vrednost *beri.Length - 1*. Dolžina niza je enaka številu vnesenih znakov, kar je v našem primeru 7. Ker se znaki štejejo od 0 dalje, ima naš zadnji znak indeks *Length - 1*. V našem primeru se spremenljivki *i* najprej priredi začetna vrednost 6. Nato se preveri pogoj. Ker je resničen *(6 >= 0)*, vstopimo v zanko in izpiše se znak na mestu *i (a)*. Nato se izvede ukaz2 *(i--)*, ki

vrednost spremenljivke *i* zmanjša za ena. Preveri se pogoj in ker še vedno velja *(5 >= 0)*, se zopet izpiše znak na mestu *5* (*d*). Na ta način se zanka *for* izvaja, dokler je vrednost v spremenljivki *i* večja ali enaka *0*. Ko je v spremenljivki vrednost *0*, je pogoj še vedno izpolnjen in v vrstici 9 se izpiše znak na mestu *0,* torej prvi znak. Sedaj spremenljivka *i* dobi vrednost *-1*. Preveri se pogoj. Ker *-1* ni večje od *0*, pogoj ni več izpolnjen. Zanka se konča in program se nadaljuje v vrstici 11. Izpiše se prazna vrsta.

Obratni niz

Kaj pa, če bi rad opravili nekoliko drugačno nalogo in iz vnesenega niza naredili obrnjeni niz nekoliko drugače. Tudi tu bomo uporabili zanko, a za razliko od prej bomo šli kar od indeksa 0 naprej. Uporabili bomo tudi stikanje nizov. Če znak dodamo nizu, se znak pretvori v ustrezni niz in niza se stakneta.

- Začnemo s praznim nizom
  - `string obrnjeniNiz = "";`
- Zanka
- Pregledamo vse znake v nizu (dolžina niza)
  - `while (i < niz.Length)`
- Dodajamo na začetek
  - `obrnjeniNiz = niz[i] + obrnjeniNiz;`

```
static void Main(string[] args)
{
      Console.WriteLine("Vnesi niz:");
      string beri = Console.ReadLine();
      Console.WriteLine("Vnešeni niz je: " + beri);
      int i = 0;
      string obrnjeniNiz = "";
      while (i < beri.Length)
      {
           obrnjeniNiz = beri[i] + obrnjeniNiz;
           i++;
      }
      Console.WriteLine("\nObrnjeni niz: " + obrnjeniNiz);
}
```

Preštej števke v nizu

Ugotovimo, koliko števk vsebuje poljuben niz:

- Preberemo niz
- Pregledamo vsak znak
  - `znak = niz[i]; // tekoči znak`
- Če je števka, povečamo števec za 1
  - `if (('0' <= znak) && (znak <= '9'))`
    `{ // če je števka`
    `koliko_stevk++; // povečanje za 1`
    `}`
- Izpišemo rezultat

```
i = 0;
while (i < dol_niza) {
    znak = niz[i]; // tekoči znak
    if (('0' <= znak) && (znak <= '9'))  {
        // če je stevka
```

```
            koliko_stevk = koliko_stevk + 1;
        }
        i = i + 1;
    }
    rezultat = "V nizu " + niz + " je " + koliko_stevk + " števk.";
```

Primerjanje nizov

Oglejmo si, kako lahko ugotovimo, da sta dva niza enaka (ang. equals)? V razredu string najdemo metodo Equals(), ki vrača rezultat tipa bool.

```
static void Main(string[] args)
{
    string niz1 = "Ana";
    string niz2 = "Zdravko";
    bool trditev = niz1.Equals(niz2);

    Console.WriteLine(trditev);
} // main
```

Program prevedemo in poženemo:

```
False
```

V programu smo uporabili metodo *Equals()*. Primerjali smo *niz1* in *niz2* . Ker sta različna, nam metoda vrne *false*.

Nize pa lahko glede enakosti primerjamo tudi z relacijskim operatorjem ==. Zgornji zgled bi torej lahko napisali kot

```
static void Main(string[] args)
{
     string niz1 = "Ana";
     string niz2 = "Zdravko";
     bool trditev = niz1 == niz2;

    Console.WriteLine(trditev);
} // main
```

Večkrat bi radi niza primerjali (ang. compare) po abecednem redu. Tako je niz "Matija" je manjši kot niz "Mojca", ker sta prva znaka enaka, drugi znak pa je v prvem nizu ('a') manjši kot v drugem nizu ('o').

Tu ne moremo uporabiti relacijskih operatorjev <, >, <=, >=. Na voljo imamo metodo `String.Compare()`, ki vrača celoštevilčno vrednost. Tako

```
String.Compare(s1,"bla")
```

vrne 0, če je niz shranjen v s1 enak nizu bla, neg. število , če je niz v s1 manjši od niza "bla" in poz. število, če je večji.

- `String.Compare("matija","mojca") // vrne negativno število`
- `bool jeManj = String.Compare(n1,n2) < 0;`

Kako si zapomniti, kako to uporabiti? Zanima nas če velja

```
niz1 <= niz2
```

(če je torej niz1 manjši (prej po abecedi) ali enak nizu niz2). Namesto zgornjega izraza, ki ga prevajalnik "ne mara", napišemo izraz

```
String.Compare(niz1,niz2) <= 0
```

Za primerjanje nizov torej uporabimo ustrezen operator in primerjamo z 0.

Poglejmo, kako deluje, oziroma, kako jo uporabljamo.

```
static void Main(string[] args)
{
     string niz1 = "Ana";
     string niz2 = "Zdravko";
     int vrednost = String.Compare(niz1,niz2);

     Console.WriteLine(vrednost);
     Console.ReadKey();
} // main
```

Program prevedemo in poženemo:

```
-1
```

Obstaja pa tudi malo drugačna metoda `CompareTo()`, ki se več ali manj razlikuje le po obliki (načinu klica). No resnici na ljubo je to "okleščena" različica metode `String.Compare()`, ki poleg tega, kar smo spoznali mi, "zna" še mnogo več. Ustrezna oblike te metode je

```
niz1.CompareTo(niz2)
```

kar ustreza klicu

```
String.Compare(niz1, niz2)
```

Prejšne zglede navedimo še v novi obliki. Tako

```
S1.CompareTo("bla")
```

vrne 0, če je niz shranjen v s1 enak nizu bla, neg. število , če je niz v s1 manjši od niza "bla" in poz. število, če je večji.

- `"matija".CompareTo("mojca") // vrne negativno število`
- `bool jeManj = n1.CompareTo(n2) < 0;`

Poglejmo še enkrat, kaj nam pove metoda `CompareTo()`. Denimo, da primerjamo *niz1* in *niz2* z `niz1.CompareTo(niz2)`. Če je niz v spremenljivki *niz1* po abecedi pred nizom, ki je v spremenljivki *niz2*, vrne metoda negativno število. Če je *niz1* enak *niz2*, vrne metoda vrednost 0. Če je *niz1* po abecedi za *niz2*, vrne metoda pozitivno število.

Metode nad nizi

Za delo z nizi poleg omenjenih metod, lastnosti in operatorja [] pogosteje uporabljamo še metode, ki so prikazane v spodnji tabeli. Da bomo imeli zbrane vse, sta v tabeli še metodi Equals in CompareTo. Vse te metode uporabljamo nad nizi. To pomeni, da jih kličemo kot

```
niz1.Metoda(niz2)
```

Metoda	Razlaga
Equals(niz)	S postopkom preverjamo enakost niza niz in niza, nad katerim uporabljamo metodo. Ukaz vrne vrednost true, če sta niza enaka.
CompareTo(niz)	S postopkom primerjamo dva niza po abecednem položaju. Če je vrnjena vrednost: • pozitivna, je niz po abecedi pred nizom, nad katerim je uporabljena metoda. • negativna, je niz po abecedi za uporabljenim nizom. • nič, sta niz in uporabljeni niz enaka (ju sestavljajo isti znaki).
ToUpper()	Metoda spremeni v nizu, nad katerim je uporabljena,vse male črke v velike.
ToLower()	Metoda spremeni v nizu, nad katerim je uporabljena, vse velike črke v male.
Substring(int,int)	Metoda vrne podniz danega niza. Prvi argument pomeni začetni položaj (indeks) v nizu. Drugi argument v javi pomeni položaj (indeks) za zadnjim znakom podniza, medtem pa v C# dolžino podniza. Če drugega argumenta ni, metoda vrne vse znake do konca niza.
IndexOf(niz)	S postopkom preverimo, če je niz podniz v nizu, nad katerim metodo uporabljamo. Metoda vrne celo število, ki predstavlja na katerem indeksu se v nizu prvič kot podniz prične niz niz. Če niz ni podniz uporabljenega niza, je vrnjena vrednost -1.

Ilustrirajmo uporabo omenjenih metod z nekaj zgledi.

Samoglasniki

Sestavimo program, ki prebere niz in ga izpiše tako, da vse samoglasnike nadomesti z zvezdico (*).

Pomagali si bomo z metodo IndexOf(), s katero bomo preverili, če je posamezen znak prebranega niza podniz niza "aAeEiIoOuU ".
Posamezne znake prebranega niza, ki jih bomo bodisi spremenili bodisi ohranili, bomo dodali v pomožni niz. Pomožni niz bo na začetku programa prazen ("").

```
C1:  static void Main(string[] args)
C2:  {
C3:       // Vnos niza
C4:       Console.Write("Vnesi niz: ");
C5:       string niz = Console.ReadLine();
C6:       // Pomožne spremenljivke
C7:       string samoglasniki = "aAeEiIoOuU"; // Niz samoglasnikov
C8:       string novNiz = ""; // Nov niz
C9:       int dolzina = niz.Length; // Dolžina prebranega niza
C10:      // Zamenjava samoglasnikov z zvezdico
C11:      for (int i = 0; i < dolzina; i++)
C12:      {
C13:          char znak = niz[i];
C14:          if (samoglasniki.IndexOf(znak) != -1)
C15:          {  // Znak je samoglasnik
```

```
C16:                novNiz = novNiz + '*'; // Samoglasnik zamenjamo z *
C17:            }
C18:            else
C19:            {
C20:                novNiz = novNiz + znak;//Ostali znaki ostanejo nespr.
C21:            }
C22:        }
C23:        // Izpis novega niza
C24:        Console.WriteLine("Spremenjen niz: " + novNiz);
C25:    }
```

Zapis na zaslonu:

```
Vnesi niz: Lepa Vida kolo vodi.
Spremenjen niz: L*p* V*d* k*l* v*d*.
```

Razlaga. Najprej preberemo niz (C5), nato deklariramo niz samoglasnikov (C7) in prazen niz (C8). Potem določimo dolžino niza niz (C9). Z zanko se sprehodimo preko vseh znakov v nizu. Znotraj zanke deklariramo spremenljivko znak in ji priredimo trenutni znak niza niz. V vrstico C14 preverimo, če je znak, katerega koda je shranjena v spremenljivki znak, podniz niza samoglasniki. Če je pogoj izpolnjen, se nizu novNiz doda znak '*' (C16), sicer pa se mu doda trenutno preverjeni znak (C20). Na koncu izvedemo izpis niza novNiz.

Galci in Rimljani

Verjetno je vsak med nami že kdaj prebral kak strip o Asterixu in Obelixu. Zato vemo, da se imena vseh Galcev zaključijo z "ix", na primer Asterix, Filix, Obelix, Dogmatix, itn. Napišimo program, ki bo prebral ime, nato pa bo izpisal "Galec!", če gre za galsko ime, v nasprotnem primeru pa bo izpisal "Rimljan!". Predpostavimo, da so vnesena imena dolga vsaj tri znake. Program zapišimo le v jeziku C#.

```
static void Main(string[] args)
{
     // Preberemo ime
     Console.Write("Vnesi ime: ");
     string ime = Console.ReadLine();
    // Podniz
    string podniz = ime.Substring(ime.Length - 2);
    podniz = podniz.ToLower(); // Da ne bo težav z velikimi črkami

    // Glede na ime izpišimo ustrezen tekst
    if(podniz.Equals("ix"))
    {
         Console.WriteLine("Galec!");
    }
    else
    {
         Console.WriteLine("Rimljan!");
    }
}
```

Kot smo povedali, metodo Equals() lahko nadomestimo z operatorjem enakosti (==). Zgornji program bi torej lahko zapisali tudi na naslednji način.

```
static void Main(string[] args)
{
    // Preberemo ime
    Console.Write("Vnesi ime: ");
    string ime = Console.ReadLine();
```

```
    // Podniz
    string podniz = ime.Substring(ime.Length - 2);
    podniz = podniz.ToLower(); // Da ne bo težav z velikimi črkami

    // Glede na ime izpišimo ustrezen tekst
    if (podniz == "ix")
    {
        Console.WriteLine("Galec!");
    }
    else
    {
        Console.WriteLine("Rimljan!");
    }
}
```

Zapis na zaslonu:

```
Vnesi ime: Filix
Galec!
```

Razlaga. Na začetku preko konzolnega okna preberemo ime osebe in ga shranimo v spremenljivki niz (C6). Potem določimo podniz iz zadnjih dveh znakov niza niz (C9). V vrstici C10 pretvorimo podniz podniz v male tiskane črke. Nato preverimo, če je podniz podniz enak nizu "ix" (C13). Če to drži, izpišemo "Galec!", sicer izpišemo "Rimljan!".

Razporeditev besed po abecedi

Napišimo program, ki bo prebral tri besede in jih izpisal po abecednem vrstnem redu. Na primer za prebrane besede buda, vescine in tempelj, naj program izpiše buda tempelj vescine. Predpostavimo, da so prebrane besede zapisane z malimi tiskanimi črkami.

Program sestavimo po korakih:
Najprej preko konzolnega okna določimo tri besede.

```
Console.Write("Vnesi prvo besedo: ");
string beseda1 = Console.ReadLine();
Console.Write("Vnesi drugo besedo: ");
string beseda2 = Console.ReadLine();
Console.Write("Vnesi tretjo besedo: ");
string beseda3 = Console.ReadLine();
```

Nato preverimo, če je beseda beseda3 abecedno pred besedo beseda1.

- Če je pogoj resničen, zamenjamo vrstni red besed beseda3 in beseda1.

```
if (beseda1.CompareTo(beseda3) > 0)
{
    string pom = beseda1;
    beseda1 = beseda3;
    beseda3 = pom;
}
```

V naslednjem koraku preverimo, če je beseda beseda2 abecedno pred besedo beseda1.

- Če je pogoj resničen, zamenjamo vrstni red besed beseda2 in beseda1.

```
if (beseda1.CompareTo(beseda2) > 0)
{
    string pom = beseda1;
    beseda1 = beseda2;
    beseda2 = pom;
```

```
}
```

Za konec preverimo še, če je beseda beseda3 abecedno pred besedo beseda2.

- Če je pogoj resničen, zamenjamo vrstni red besed beseda3 in beseda2.

```
if (beseda2.CompareTo(beseda3) > 0)
{
    string pom = beseda2;
    beseda2 = beseda3;
    beseda3 = pom;
}
```

Besede so abecedno urejene, zato jih izpišemo. Ločimo jih s presledkom.

```
Console.WriteLine("\n" + beseda1 + " " + beseda2 + " " + beseda3);
```

Združimo posamezne korake v celoten program.

```
static void Main(string[] args)
{
    // Vnesene besede
    Console.Write("Vnesi prvo besedo: ");
    string beseda1 = Console.ReadLine();
    Console.Write("Vnesi drugo besedo: ");
    string beseda2 = Console.ReadLine();
    Console.Write("Vnesi tretjo besedo: ");
    string beseda3 = Console.ReadLine();

    // beseda3 je abecedno pred beseda1
    if(beseda1.CompareTo(beseda3) > 0)
    {
        string pom = beseda1;
        beseda1 = beseda3;
        beseda3 = pom;
    }

    // beseda2 je abecedno pred beseda1
    if(beseda1.CompareTo(beseda2) > 0)
    {
       string pom = beseda1;
       beseda1 = beseda2;
       beseda2 = pom;
    }

    // beseda3 je abecedno pred beseda2
    if(beseda2.CompareTo(beseda3) > 0)
    {
       string pom = beseda2;
       beseda2 = beseda3;
       beseda3 = pom;
    }

    // Izpis po abecedi
    Console.WriteLine("\n" + beseda1 + " " + beseda2 + " " + beseda3);
}
```

Zapis na zaslonu:

```
Vnesi prvo besedo: buda
Vnesi drugo besedo: vescina
Vnesi tretjo besedo: tempelj

buda tempelj vescina
```

Še nekaj metod

Povzemimo še vse skupaj v tabelo in dodajmo še nekaj uporabnejših metod.

Indeks	**Razlaga**
[indeks]	Dostop do znaka na določeni poziciji.
Lastnost	**Razlaga**
Length	Število znakov nizu.
Metoda	**Razlaga**
StartsWith(string)	Vrne logično vrednost, ki označuje, ali se nek niz začenja z navedenim nizom.
EndsWith(string)	Vrne logično vrednost, ki označuje, ali se nek niz končuje z navedenim nizom.
IndexOf(niz)	S postopkom preverimo, če je niz podniz v nizu, nad katerim metodo uporabljamo. Metoda vrne celo število, ki predstavlja na katerem indeksu se v nizu prvič kot podniz prične niz niz. Če niz ni podniz uporabljenega niza, je vrnjena vrednost -1.
IndexOf(string, začetni indeks)	Vrne celo število ki predstavlja mesto (indeks), kjer se navedeni (pod)niz v nekem nizu od začetnega indeksa naprej prvič pojavi. Če navedenega (pod)niza ni, je vrnjena vrednost -1.
Insert(začetni indeks, string)	Vrne niz, v katerega je na navedeno mesto (začetni indeks) vrinjen navedeni niz.
Equals(niz)	S postopkom preverjamo enakost niza niz in niza, nad katerim uporabljamo metodo. Ukaz vrne vrednost true, če sta niza enaka.
CompareTo(niz)	S postopkom primerjamo dva niza po abecednem položaju. Če je vrnjena vrednost: - pozitivna, je niz po abecedi pred nizom, nad katerim je uporabljena metoda. - negativna, je niz po abecedi za uporabljenim nizom. - nič, sta niz in uporabljeni niz enaka (ju sestavljajo isti znaki).
ToUpper()	Metoda spremeni v nizu, nad katerim je uporabljena,vse male črke v velike.
ToLower()	Metoda spremeni v nizu, nad katerim je uporabljena, vse velike črke v male.
Substring(začetni_indeks)	Vrne del niza, ki se začne na navedenem mestu in vsebuje vse znake do konca niza.
Substring(začetni_indeks, dolžina)	Vrne del niza, ki se začne na navedenem mestu in ima navedeno dolžino.
PadLeft(skupna_dolžina)	Vrne niz, ki je DESNO poravnan in na levi strani zapolnjen s tolikšnim številom presledkov, da je skupno število znakov enako vrednosti skupna_dolžina.

PadRight(skupna_dolžina)	Vrne niz, ki je LEVO poravnan in na desni strani zapolnjen s tolikšnim številom presledkov, da je skupno število znakov enako vrednosti skupna_dolžina.
Remove(začetni_indeks, N)	Vrne niz, iz katerega je odstranjeno N znakov od mesta začetni_indeks naprej.
Replace(staristring, novistring)	Vrne niz, v katerem je na vsakem mestu, kjer je bil v starem nizu (pod)niz stariniz, sedaj (pod)niz noviniz.
Trim()	Vrne niz iz katerega so odstranjeni vsi vodilni in končni presledki.

Nekaj primerov:

```
//dostop do posameznega znaka v nizu
string znaki = "abcdefg";
char a=znaki[0];  // 'a'
char b=znaki[1];  // 'b'

string beseda="Danes je lep dan!";
//sestavimo nov niz iz nekaterih znakov niza beseda
string znakiInPresledki = beseda[0]+ beseda[3]+ beseda[4]+ beseda[14];
//znakiInPresledki dobi vrednost "Desa"

//Uporaba metod StartsWith in EndsWith
bool zacneZabc = znaki.StartsWith("abc"); // true
bool koncaZabc = znaki.EndsWith("abc");   // false

//Uporaba metode IndexOf
string sola = "Tehniški Šolski Center Kranj";
int index1 = sola.IndexOf(" ");// 8, ker se string " " pojavi prvič na
                               // osmem mestu
int index2 = sola.IndexOf(' '); // 8, ker se znak ' ' pojavi prvič na osmem
                                // mestu
int index3 = sola.IndexOf("Center");// 16, ker se string "Center" pojavi
                                   // prvič na 16 mestu
int index4 = sola.LastIndexOf(" ");// 22, ker se string " " pojavi zadnjič
                                   // na 22 mestu

//Uporaba metod Remove, Insert in Replace
sola = sola.Remove(0, 9);  // Šolski center Kranj
sola = sola.Insert(sola.Length, ", Slovenija");// Šolski center Kranj
                                               //, Slovenija
sola = sola.Replace("Slovenija", "4000 Kranj");// Šolski center Kranj, 4000
                                               // Kranj

//Uporaba metod Substring, ToUpper in ToLower
string ime = "aNJA";
string prvaCrka = ime.Substring(0, 1).ToUpper(); // A
string drugeCrke = ime.Substring(1).ToLower();  // nja
ime = prvaCrka + drugeCrke;                      // Anja

//Kopiranje enga stringa v drug string
string s1 = "abc";
string s2 = s1; // string s2 dobi vrednost s1, torej s2 postane "abc"
s2 = "def";     // string s2 postane "def", string s1 pa se ne spremeni
string s3 = s1 + s2;  // strig s3 dobi vrednost "abcdef"
```

```
//Uporaba metode Substring
string polnoIme = "  Edward C Koop     ";      // "  Edward C Koop     "
polnoIme = polnoIme.Trim();                    // "Edward C Koop"
int prvipresledek = polnoIme.IndexOf(" ");    // 6
string lastnoIme = polnoIme.Substring(0,prvipresledek); // "Edward"

string naslov = " |34 Kališka ulica|Slovenija|Kranj|4000 ";
naslov = naslov.Trim();     // "|34 Kališka ulica|Slovenija|Kranj|4000"
naslov.Remove(0,1);         //iz naslova odstranimo prvi znak
string naslov1= naslov.Remove(naslov.Length-1,1);// iz naslova odstranimo
                                                 // zadnji znak

int indeksUlice = naslov.IndexOf("|") + 1;  // 1
int indeksDrzave = naslov.IndexOf("|", indeksUlice) + 1;  // 18
int indeksKraja = naslov.IndexOf("|", indeksDrzave) + 1;  // 28
int indeksPoste = naslov.IndexOf("|", indeksKraja) + 1;   // 34

string ulica = naslov.Substring(0, indeksDrzave-1); // 34 Kališka ulica
string mesto = naslov.Substring(indeksKraja,indeksPoste - indeksKraja - 1);
// Kranj
string posta = naslov.Substring(indeksPoste); // 4000

//Uporaba metode Insert
string telefonskaStevilka = "041827919";                    // "041827919"
telefonskaStevilka = telefonskaStevilka.Insert(3, "-");    // "041-827919"
telefonskaStevilka = telefonskaStevilka.Insert(7, "-");    // "041-827-919"

//Uporaba metode Replace
string datum = "21-08-2007";     // "21-08-2007"
datum = datum.Replace("-", "/");
```

Naloge za utrjevanje znanja iz nizov

Kaj izpišejo spodnji deli programov :

- ```
  string niz = "Tantadruj";
  char y;
  y = niz[5];
  Console.WriteLine("V nizu je crka " + y);
  ```

- ```
  string niz1 = "Tantadruj";
  int n = niz1.Length;
  string niz2 = "";
  int i = 0;
  while(i < n) {
      char x = niz1[i];
      niz2 = x + niz2;
      i = i + 1;
  }
  Console.WriteLine(niz2);
  ```

- ```
  string niz1 = "Tantadruj";
  int n = niz1.Length;
  string niz2 = "";
  ```
  ```

```
int i = 3;
while(i < n) {
    niz2 = niz2 + niz1[i];
    i = i + 1;
}
niz2 = niz2 + niz1[0] + niz1[1] + niz1[2];
Console.WriteLine(niz2);
```

_____

- Kaj izpiše naslednja metoda:

```
public static void Izpis (){
  string a = "moje bojno polje";
  System.Console.Write(a.IndexOf("oj"));
}
```

a) 6
b) 7
c) 1
d) 2
e) nič od navedenega, ampak _____

- Dani niz izpiši obrnjeno.

Primera:

-

"1. januar 2008"
Niz 1. januar 2008 izpisan obrnjeno je 8002 raunaj .1

"Lahek izpit"
Niz Lahek izpit izpisan obrnjeno je tipzi kehaL

- Na svetovnem spletu uporabljamo več kodnih tabel znakov, od katerih samo nekatere vsebujejo šumevce (č, š, ž). Da se šumevci ne pretvorijo v nerazumljive znake, jih pretvorimo v ustrezne sičnike (c, s, z) in jih take tudi izpišimo. Napišite program, ki prebere vrstico besedila s šumevci in na zaslon izpiše spremenjeno besedilo.

- Sestavite program, ki prebere niz in ga izpiše tako, da med znake besede vrine presledke.

Primer: Prebran niz je Lepa Anka kolo vodi.
L e p a A n k a k o l o v o d i .

- Napišite program Okvir, ki bo zahteval vnos niza. Nato ga bo izpisal na zaslon v "okvirju".

Primer:

Vnesi niz: `Mladost je norost, skače čez jarke, kjer je most.`

```
*-----------------------------------------------*
|Mladost je norost, skače čez jarke, kjer je most.|
*-----------------------------------------------*
```

- Sestavite program, ki bo zahteval vnos besede. Nato to besedo izpiše na zaslon tako, da bo beseda izpisana v obliki trikotnika.

Primer: Vnesena beseda je JEZIK

```
JEZIK
JEZI
```

```
JEZ
JE
J
```

- Napiši program, ki bo prebral niz in preveril prvi znak. Če prvi znak ni črka, naj izpiše "Niz se ne začne s črko." Če je prvi znak velika črka, naj izpiše "Niz niz ima veliko začetnico.". Če je prvi znak mala črka, naj niz spremeni, tako da bo prvi znak velika začetnica in izpiše: "Vneseni niz niz ni imel velike začetnice. Po popravku zgleda tako: popravljen niz niz.".

- Dana je deklaracija string stavek = "moj stavek"; S pomočjo lastnosti Length ter metode ToUpper spremeni prvo in zadnjo črko tega niza v veliko črko.

- Ugotovi na katerem mestu spremenljivke stavek tipa string se prvič pojavi veznik in (uporabi metodo IndexOf)?

- Iz spremenljivke stavek tipa string odstrani vse znake od 10. znaka naprej! Uporabi metodo Remove!

- Dana je spremenljivka stavek tipa string, spremenljivka ima že neko vrednost (vsebuje več kot 10 znakov).
  - Odstrani vse vodilne in končne presledke;
  - Vse črke v stringu stavek spremeni v velike črke angleške abecede;
  - Ugotovi in izpiši prvi in zadnji znak tega stringa;
  - Ugotovi in izpiši koliko znakov je v tem stringu;
  - V spremenljivko beseda (string) zapiši prvih pet znakov stringa stavek;
  - Zadnje tri znake tega stringa nadomesti s pikicami;
  - Na sredino stringa stavek vrini tri podčrtaje;
  - Iz stringa stavek odstrani vse znake od šestega znaka naprej.

- V nizu stavek vse številke nadomesti z besednim opisom (številko 1 zamenjaj z ena, številko 2 z dva, …)

- Dane so tri stpremenljivke tipa string st1, st2 in st3, ki so že inicializirani. Deklariraj spremenljivko st4 tipa string, katere vrednost naj bo sestavljena iz zadnjih znakov nizov v st1, st2 in st3!

- Dan je zelo dolg niz, sestavljen iz številk. Ugotovi, katerih številk je v nizu največ.

- Izpiši najprej vse samoglasnike poljubnega stavka, nato ta stavek izpiši navpično, vsak znak v svojo vrsto!

- Napiši program, ki prebere niz in vsak znak niza razmnoži tolikokrat kot mu naroči uporabnik z vnesenim številskim parametrom. Primer izpisa: Uporabnik je vnesel niz AVTO in zahteval, da se vsak znak ponovi trikrat.
  AAAVVVTTTOOO

- Napiši program, ki bo uporabniku omogočal vpis poljubnega stavka. Iz vpisanega stavka program nato izloči samoglasnike. Namesto ostalih znakov pa izpiše podčrtaj (_).

- Preberi poljuben stavek in ga izpiši navpično in nato še diagonalno (vsaka črka v svoji vrstici, za en znak bolj v desno!

- Napiši program, ki vse šumnike š, č in ž nekega stavka nadomesti z znaki s, c in z!

- Preberi poljuben string. Ugotovi in izpiši, koliko presledkov vsebuje.

- Preberi kemijsko formulo in jo izpiši tako, da so številke vrstico nižje. Če npr. prebereš formulo H2SO4, jo izpiši kot
  H SO
  2 4

- Dana sta dva niza, ime in priimek. Sestavi nov niz sifra tako, da najprej obrneš ime in priimek, nato pa izmenično iz zaporednih črk obrnjenega imena in priimka sestaviš nov niz. Iz nizov ime in priimek vzemi le toliko črk, kot znaša dolžina krajšega od obeh nizov.

- Beri znake toliko časa, da je vneseni znak enak pika (.). Prebrane znake nato izpiši v obliki stavka.

- Sestavi preprost program, ki bo kodiral in dekodiral kratka telefonska sporočila (sms-e). Program mora v obrniti vrstni red besed v sporočilu in vrstni red črk v besedi.

- Preberi 5 stavkov. Če je v stavku manj kot 5 znakov, ga ne upoštevaj (stavek continue). Na koncu ugotovi in izpiši najdaljši stavek.

Tabele

Pogosto se srečamo z večjo količino podatkov istega tipa, nad katerimi želimo izvajati podobne (ali enake) operacije. Takrat si pomagamo s tabelami. Tabelo ali polje (ang. *array*) v jeziku C# uporabljamo torej v primerih, ko moramo na enak način obdelati skupino vrednosti istega tipa. Oglejmo si tak zgled. Tabelarične spremenljivke imajo vse enako ime (npr. tabela), razlikujejo pa se po **indeksu**, zaradi česar vsako od njih obravnavamo kot samostojno spremenljivko.

Preberi 5 števil in jih izpiši v obratnem vrstnem redu

Denimo, da je naloga *Preberi 5 števil in jih izpiši v obratnem vrstnem redu*. Vsa števila moramo prebrati (shraniti), ker jih bomo potrebovali šele, ko bodo prebrana vsa. Ustrezen bo na primer tak program:

```
static void Main(string[] args)
{
     // branje
     Console.WriteLine("Vnesi število:");
     string beri = Console.ReadLine();
     int x1 = int.Parse(beri);
     Console.WriteLine("Vnesi število:");
     beri = Console.ReadLine();
     int x2 = int.Parse(beri);
     Console.WriteLine("Vnesi število:");
     beri = Console.ReadLine();
     int x3 = int.Parse(beri);
     Console.WriteLine("Vnesi število:");
     beri = Console.ReadLine();
     int x4 = int.Parse(beri);
     Console.WriteLine("Vnesi število:");
     beri = Console.ReadLine();
     int x5 = int.Parse(beri);
     // izpis
     Console.WriteLine("Števila v obratnem vrstnem redu: ");
     Console.WriteLine(x5);
     Console.WriteLine(x4);
     Console.WriteLine(x3);
     Console.WriteLine(x2);
     Console.WriteLine(x1);
}
```

Če malo pomislimo, vidimo, da v našem program prazzaprav ponavljamo dva sklopa ukazov. Prvi je

```
Console.WriteLine("Vnesi število:");
beri = Console.ReadLine();
int x2 = int.Parse(beri);
```

in drugi

```
Console.WriteLine(x3);
```

Vsak sklop ponovimo 5x. Razlikujejo se le po tem da v njih nastopa enkrat x2, drugič x3, tretjič x4 … To nam da misliti, da bi lahko uporabili zanko:

zanka
    preberi i-to število
    shrani ga v xi
    Povečaj i za 1
Naj bo i = prebrano število števil
zanka
    izpiši xi
    zmanjšaj i za 1

```
static void Main(string[] args)
{
    // branje
    string beri;
    for (int i = 1; i <= 50; i++)
    {
        Console.WriteLine("Vnesi število:");
        beri = Console.ReadLine();
        int xi = int.Parse(beri);
    }
    for (int i = 50; i >= 1; i--)
        Console.WriteLine(xi);
}
```

Prevajalniku tisti xi ni najbolj všeč in v drugi zanki pravi, da "ne obstaja". No razlog je v tem, da so tisti naši xi deklarirani v prvi zanki in jih "drugje ni". Torej jih moramo deklarirati zunaj.

Spremenimo v

```
static void Main(string[] args)
{
    // branje
    string beri;
    int x1, x2, x3, x4, x5;
    for (int i = 1; i <= 5; i++)
    {
        Console.WriteLine("Vnesi število:");
        beri = Console.ReadLine();
        xi = int.Parse(beri);
    }
    for (int i = 5; i >= 1; i--)
        Console.WriteLine(xi);
}
```

A prevajalnik pravi, da ne pozna spremenljivke xi? No, morda pa bi bilo dobro deklarirati še to in dodamo

```
int xi;
```

A ko poženemo program, le ta 5x izpiše zadnje vnešeno število?

Indeksi

Mi bi v prejšnjem zgledu želeli, da xi pomeni x1, x2, x3, ... (glede na vrednost i). Prevajalnik pa trmasto vztraja, da je to spremenljivka z imenom xi (torej ena sama).
Kako torej napišemo indekse? Indeks napišemo za imenom tabele med oglatimi oklepaji ([]). Tako zapis igralci[4] pomeni element z indeksom 4 tabele igralci.

Torej bi prejšnji zgled morali napisati kot

```
static void Main(string[] args)
{
    // branje
    string beri;
    int x1, x2, x3, x4, x5;
    for (int i = 1; i <= 5; i++)
    {
        Console.WriteLine("Vnesi število:");
        beri = Console.ReadLine();
        x[i] = int.Parse(beri);
    }
    for (int i = 5; i >= 1; i--)
        Console.WriteLine(x[i]);
}
```

Vendar žal prevajalnik še vedno ni zadovoljen. Namreč pozna le spremenljike x1, x2, ... ne ve pa, da je x ime tabele, kjer bi radi imeli 5 indeksov (torej prostor za 5 števil).

Deklaracija tabele

Prvi korak pri ustvarjanju tabele je najava le te. To storimo tako, da za tipom tabele navedemo oglate oklepaje in ime tabele.

```
podatkovniTip[] imeTabele;
```

Z zgornjim stavkom zgolj napovemo spremenljivko, ki bo hranila naslov bodoče tabele, ne zasedemo pa še nobene pomnilniške lokacije, kjer bodo elementi tabele dejansko shranjeni. Potrebno količino zagotovimo in s tem tabelo dokončno pripravimo z ukazom *new*:

```
imeTabele = new podatkovniTip[velikost];
```

Ukaz *new* zasede dovolj pomnilnika, da vanj lahko shranimo *dolzina* spremenljivk ustreznega tipa in vrne njegov naslov. Velikost tabele je enaka številu elementov, ki jih lahko hranimo v tabeli.

Napoved in zasedanje pomnilniške lokacije lahko združimo v en stavek:

```
podatkovniTip[] imeTabele = new podatkovniTip[velikost];
```

Primeri najave tabele:

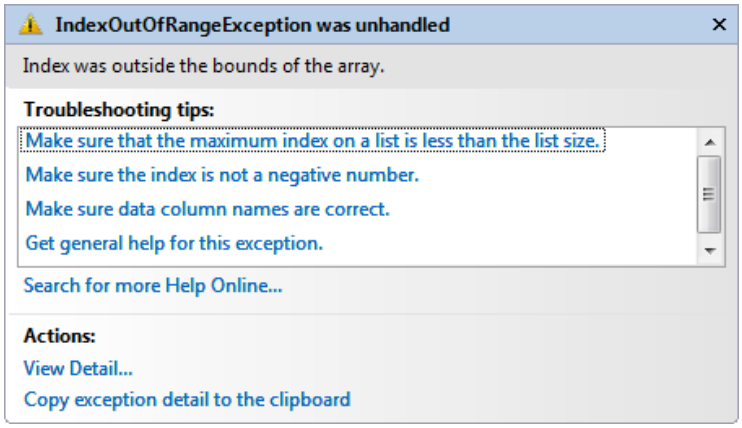
```
// tabela 10 celih števil
int[] stevila = new int[10];
// tabela 3 realnih števil
double[] cena = new double[3];
// tabela 4 znakov
char[] tabelaZnakov = new char[4];
// tabela 500 nizov
string[] ime = new string[500];
```

Sedaj lahko končno naš zgled napišemo prav.

```
static void Main(string[] args)
{
```

```
    // branje
    string beri;
    int[] x = new int[5];
    for (int i = 1; i <= 5; i++)
    {
        Console.WriteLine("Vnesi število:");
        beri = Console.ReadLine();
        x[i] = int.Parse(beri);
    }
    for (int i = 5; i >= 1; i--)
        Console.WriteLine(x[i]);
}
```

A med izvajanjem se program "sesuje" s sporočilom



Zakaj? Grafično si lahko enodimenzionalno tabelo predstavljamo takole:

```
int[] tab = new int[5];//ime tabelarične spremenljivke je tab
```

tab[0]	tab[1]	tab[2]	tab[3]	tab[4]

```
tab[4] = 100;  // peti element v tabeli ( indeksi se označujejo od 0 naprej !!!)
```

ime tabele (polja) indeks vrednost

Vsaka tabelarična spremenljivka ima svoj indeks, preko katerega ji določimo vrednost (jo inicializiramo) ali pa jo uporabljamo tako kot običajno spremenljivko. Začetni indeks je enak nič (0) končni indeks pa je za ena manjši kot je dimenzija tabele (dimenzija – 1).

S posameznimi elementi tabele lahko operiramo enako kot s spremenljivkami tega tipa. Tako je v spodnjem zgledu npr. *tabela[4]* povsem običajna spremenljivka tipa *int*.

Primer:

```
// tabelo napolnimo z vrednostmi
tabela[0] = 10;
tabela[1] = 20;
tabela[2] = 31;
```

```
tabela[3] = 74;
tabela[4] = 97;
tabela[5] = 31;
```

Posameznim tabelaričnim spremenljivkam (komponentam) lahko prirejamo vrednost, ali pa jih uporabljamo v izrazih, npr.:

```
int[] tab = new int[5];
tab[0] = 100;
tab[1] = tab[0]-20;      // tab[1] dobi vrednost 80
tab[2] = 300;
tab[3] = 400;
tab[4] = tab[1] + tab[3]; // tab[4] dobi vrednost 500
```

Izgled tabele po izvedbi zgornjih stavkov je torej takle:

100	80	300	400	380

Pri uporabi indeksov pa moramo paziti, da ne zaidemo izven predpisanih meja. In ker smo v našem primeru uporabljali x[5] (ko je bil i == 5), smo dobili napako IndexOutOfRangeException, ki vedno pomeni, da smo zašli izven tabele.

Če sedaj upoštevamo povedano, vidimo, da bomo namesto indeksov 1, 2, 3, 4 in 5 uporabili indekse 0, 1, 2, 3, in 4. S programom

```
static void Main(string[] args)
{
    // branje
    string beri;
    int[] x = new int[5];
    for (int i = 0; i <= 4; i++)
    {
        Console.WriteLine("Vnesi število:");
        beri = Console.ReadLine();
        x[i] = int.Parse(beri);
    }
    for (int i = 4; i >= 0; i--)
        Console.WriteLine(x[i]);
}
```

pa končno dobimo željeni rezultat!

Najbolj uporabna lastnost tabel je ta, da jih zlahka uporabljamo v zankah. S spreminjanjem indeksov v zanki poskrbimo, da se operacije izvedejo nad vsemi elementi tabele.

Deklarirajmo tabelo 100 celih števil in jo inicializirajmo tako, da bodo imeli vsi elementi tabele vrednost 1:

```
int[] tabela = new int[100];

for (int i = 0; i < 100; i++)
{
    tabela[i] = 1;

}
```

Velikost tabele lahko določimo med samim delovanjem programa. Dimenzijo tabele torej lahko določi uporabnik, glede na svoje potrebe:

```
Console.Write("Določi dimenzijo tabele: ");
int dimenzija = int.Parse(Console.ReadLine());
    //Dimenzijo tabele določi uporabnik med izvajanjem!!!

int[] tabela = new int[dimenzija];
```

Sočasno z najavo lahko elementom določimo tudi začetne vrednosti. Te zapišemo med zavita oklepaja. Posamezne vrednosti so ločene z vejico.

Primer:

```
int[] stevila = {1, 2, 3, 6, 8, 12, 14, 4, 7, 9};
```

Ob naštevanju vrednosti ne moremo navesti dolžine tabele, saj jo prevajalnik izračuna sam. Paziti moramo, da vsi elementi ustrezajo izbranemu tipu.

Povzemimo vse načine deklaracije tabele:

1. način: Najprej najavimo ime tabele. Nato z operatorjem new dejansko ustvarimo tabelo.

Primer:

```
int[] tabela = new int[5]; // ustvarimo tabelo dolžine 5
int[] tab1; // Vemo, da je tab1 ime tabele celih števil,
// tabela kot taka pa še ne obstaja
tab1 = new int[20]; // tab1 kaže na tabelo 20 celih števil
```

Seveda ni nujno, da si ukaza sledita neposredno drug za drugim. Prav tako ni nujno, da za določanje velikosti uporabimo konstanton. Napišemo lahko poljubni izraz.

```
tab1 = new int[2 + 5]; // tab1 kaže na tabelo 7 celih števil
```

kjer lahko nastopajo tudi spremenljivke, ki imajo v tistem trenutku že prirejeno vrednost

```
int vel = 10;
tab1 = new int[2 * vel]; // tab1 kaže na tabelo 20 celih števil
```

ki smo jo seveda lahko tudi prebrali

```
int[] tab = new int[int.Parse(Console.ReadLine())];
```

No, omenjeno "kolobocijo" zgoraj raje napišimo kot

```
string beri = Console.ReadLine();
int velTab = int.Parse(beri);
int[] tab = new int[velTab];
```

2. način: Oba zgornja koraka združimo v enega.

Primera:

```
int[] tab1 = new int[2]; // Vsi elementi so samodejno nastavljeni na 0
double[]tab2 = new double[5];//Vsi elementi so samodejno nastavljeni na 0.0
```

Tako v 1. in 2. načinu velja, da se, ko z new ustvarimo tabelo, vsi elementi samodejno nastavijo na začetno vrednost, ki jo določa tip tabele (npr. int - 0, double - 0.0, bool - false).

3. način: Tabelo najprej najavimo, z operatorjem new zasedemo pomnilniško lokacijo, nato pa elementom določimo začetno vrednost.

Primer:

```
int[] tab3 = new int[] { -7, 5, 65 }; // Elementi so -7, 5 in 65
```

lahko pa smo tudi "pridni" in velikost povemo še sami

Primer:

```
int[] tab3 = new int[3] { -7, 5, 65 }; // Elementi so -7, 5 in 65
```

4. način: Tabelo najprej najavimo, nato pa elementom določimo začetno vrednost. Velikosti tabele ni potrebno podati, saj jo prevajalnik izračuna sam.

Primer:

```
char[] tab4 = { 'a', 'b', 'c' }; // Elementi so 'a', 'b', in 'c'
```

Omenimo še enkrat, da dolžino tabele *stevila* (število elementov, ki jih lahko shranimo v tabelo) dobimo z izrazom *stevila.Length*.

Indeksi v nizih in tabelah

Izraz, s katerim dostopamo do i-tega elementa tabele (imeTabele[i]), se v jeziku C# po videzu ujema z izrazom za dostopanje do i-tega znaka v nizu (imeNiza[i]). Izraza pa se ne ujemata v uporabi. Pri tabelah uporabljamo izraz za pridobitev in spremembo elementa tabele, medtem, ko ga pri nizih uporabljamo le za pridobitev znaka. V primeru, da izraz uporabimo za spremembo določenega znaka v nizu, nam prevajalnik vrne napako.

```
C:\Documents and Settings\Administrator\Desktop\VajeC#\Test12\Test12\
Program.cs(10,13): error CS0200: Property or indexer 'string.this[int]' cannot be
assigned to -- it is read only
```

Primer:

```
// Deklaracija tabele in niza
int[] tab = { 1, 6, 40, 15 };
string niz = "Trubarjevo leto 2008.";

// Pridobivanje
int el = tab[2]; // Vrednost spremenljivke je 6.
char z = niz[2]; // Vrednost spremenljivke je koda zanka 'u'.

// Spreminjanje
tab[1] = 5; // Spremenjena tabela je [1, 5, 40, 15].
niz[18] = '9'; // Izpiše se napaka.
```

Poglejmo si preprost primer tabele imen. Napolnimo tabelo z imeni in izpišimo njeno dolžino.

```
1: static void Main(string[] args)
2: {
3:     string[] imena={"Jan", "Matej", "Ana", "Grega", "Sanja"};
4:     Console.WriteLine("Dolzina tabele je " + imena.Length + ".");
```

```
5:      Console.WriteLine("Ime na sredini tabele je  " +
6:                        imena[imena.Length / 2] + ".");
7: } // main
```

Program prevedemo in poženemo:

```
Dolzina tabele je 5.
Ime na sredini tabele je  Ana.
```

Opis programa
V 3. vrstici smo najavili tabelo *imena* tipa *string[]* in jo napolnili z elementi. Zatem smo izpisali njeno dolžino z ukazom *imena.length* . V 4. vrstici želimo izpisati element, ki se nahaja na sredini naše tabele. Izraz *imena.length / 2* nam vrne *2*, element, ki se nahaja na mestu *imena[2]* je *Ana.*

Podrobneje si poglejmo, kako indeksiramo podatke v tabeli *imena.*

Prvi element tabele ima indeks *0* in vsebuje niz *"Jan"*, drugi ima indeks *1* in vsebuje niz *"Matej"*, tretji ima indeks *2* in vsebuje niz *"Ana"*, četrti ima indeks *3* in vsebuje niz *"Grega"* in peti ima indeks *4* in vsebuje niz *"Sanja"*. Tabelo *imena* si lahko predstavljamo takole:

indeks	*0*	*1*	*2*	*3*	*4*
vrednost	Jan	Matej	Ana	Grega	Sanja

Neujemanje tipov pri deklaraciji

Kaj se bo zgodilo, če napišemo naslednji program.

```
1:  static void Main(string[] args)
2:  {
3:    string[] stevila = { 1, 2, 3 };
4:    Console.WriteLine("Dolžina tabele je " +
5:                      stevila.Length + ".");
6:  }
```

Program se ne prevede. Če pogledamo vrstico 3, smo deklarirali tabelo nizov, tej tabeli pa smo priredili celoštevilske vrednosti. Ker se tipa ne ujemata, bo prevajalnik javil:

Error 3 Cannot implicitly convert type 'int' to 'string'

Napako popravimo tako, da napišemo ustrezen (pravilen) tip.

```
1:  static void Main(string[] args)
2:  {
3:    int[] stevila = { 1, 2, 3 };
4:    Console.WriteLine("Dolzina tabele je " +stevila.Length + ".");
5:  }
```

Program prevedemo in poženemo:

```
Dolzina tabele je 3.
```

Izpis tabele

Poskusimo izpisati vse elemente tabele *imena* na naslednji način.

```
static void Main(string[] args)
{
     string[] imena={"Jan", "Matej", "Ana", "Grega", "Sanja"};

     Console.WriteLine(imena);
} // main
```

Program prevedemo in poženemo:

```
System.String[]
```

Dobimo čuden izpis. V spremenljivki *imena* namreč ne hranimo tabele imen, ampak le naslov, kje se tabela nahaja. Z ukazom *Console.WriteLine(imena)* izpišemo naslov, kjer se nahaja naša tabela, in ne posameznih elementov. Če želimo izpisati vrednosti slednjih, jih moramo obravnavati vsakega posebej. Omenili smo že, da do *i*–tega elementa (natančneje, do elementa z indeksom i) dostopamo z izrazom *imeTabele[i]*. Popravimo program. Za izpis vseh elementov tabele bomo uporabili zanko *for*,

```
static void Main(string[] args) {
    string[] imena={"Jaka", "Matej", "Ana", "Grega", "Sanja"};

    for(int i = 0; i < imena.Length; i++)
      Console.WriteLine(imena[i]);
} // main
```

Program prevedemo in poženemo:

```
Jaka
Matej
Ana
Grega
Sanja
```

Opis programa:
Najprej definiramo tabelo tipa *string* in jo napolnimo s podatki. Njeno vsebino bomo izpisali na ekran s pomočjo zanke *for*. Definiramo spremenljivko *i* in ji priredimo vrednost *0* , zatem preverimo če je *i* manjše od dolžine tabele, ki je v našem primeru *5* (pogoj je izpolnjen), izpišemo spremenljivko, ki se nahaja na mestu *imena[0]* to je *Jaka*. Vrednost *i* povečamo za ena in ponovno preverimo resničnost pogoja (*i < imena.Length*), ker je pogoj ponovno resničen izpišemo vrednost spremenljivke *imena[1] Matej* , …postopek nadaljujemo toliko časa, da izpišemo še preostal del tabele, ko pogoj ni več izpolnjen (izpisali smo vse vrednosti tabele) končamo.

Poskusimo izpisati peti element tabele *imena*.

```
static void Main(string[] args) {
    string[] imena={"Jaka", "Matej", "Ana", "Grega", "Sanja"};

    Console.WriteLine(imena[5]);
} // main
```

Program prevedemo in poženemo:

Prevajalnik javi napako, saj smo poskušali poklicati element, ki je izven definirane meje. Kot smo že povedali, se veljavni indeksi vedno gibljejo v meji od *0* do *dolzina – 1*, v našem primeru od *0* do *4*.

Primerjanje tabel

Oglejmo si še en primer pogoste napake. Denimo, da želimo primerjati dve tabeli. Zanima nas, če so vsi istoležni elementi enaki. "Naivna" rešitev z t1 == t2 nam ne vrne pričakovan rezultat.

```
static void Main(string[] args)
{
    int[] t1 = new int[] { 1, 2, 3 };
    int[] t2 = new int[] { 1, 2, 3 };
    Console.WriteLine(t1 == t2);
} //main
```

Program prevedemo in poženemo:

```
False
```

Opis programa.
Čeprav obe tabeli vsebujeta enake elemente, nam izpis pove, da tabeli nista enaki. Zakaj? Spomnimo se, da *t1* in *t2* vsebujeta samo naslov, kjer se nahajata tabeli. Ker sta to dve različni tabeli (sicer z enakimi elementi, a vseeno gre za dve tabeli), zato tudi naslova nista enaka. In to nam pove primerjava *t1 == t2*.

Oglejmo si, kako bi pravilno primerjali dve tabeli.

```
1: static void Main(string[] args)
2: {
3:         int[] t1 = new int[] {1, 2, 3};
4:         int[] t2 = new int[] {1, 2, 3};
5:
6:         bool je_enaka = true;
7:
8:         if (t1.Length == t2.Length)
9:         {
10:            for(int i = 0; i < t1.Length; i++)
11:            {
12:                 if (t1[i] != t2[i])
13:                    je_enaka = false;
14:            }
15:         }
16:         else je_enaka = false;
17:
18:         if (je_enaka)
19:          Console.WriteLine("Tabeli sta enaki.");
20:         else Console.WriteLine("Tabeli nista enaki.");
21: } //main
```

Opis programa.
V 8. vrstici najprej preverimo če sta tabeli enako veliki. Če je ta pogoj izpolnjen, nadaljujemo s primerjavo elementov znotraj tabele. Z zanko for se sprehodimo po vseh elementih in na vsakem koraku primerjamo *t1[0] != t2[0], t1[1] != t2[1], …* Če bi naleteli vsaj na en par neenakih števil, bi spremenljivko *je_enaka* nastavili na *false,* saj tabeli potem nista enaki. V našem primeru pa je ta pogoj vedno neresničen, saj so *1 != 1, 2 != 2, …* neresnične izjave, zato izpišemo *Tabeli sta enaki.* Če pa naletimo na tabeli, ki nista enako veliki ali pa je pogoj v 12. vrstici resničen vsaj enkrat, se izpiše *Tabeli nista enaki.*

Zgledi

Dnevi v tednu

Deklariraj tabelo sedmih nizov in jo inicializiraj tako, da bo vsebovala dneve v tednu.
Tabelo nato še izpiši, vsak element tabele v svojo vrsto.

```
string[]  dneviVTednu  =  new  string[7]  {"Ponedeljek",  "Torek",  "Sreda",
                           "Četrtek", "Petek", "Sobota", "Nedelja" };
for (int i=0;i<7;i++)
{
     Console.WriteLine(dneviVTednu[i]);
}
```

Mrzli meseci

Preberi povprečne temperature v vseh mesecih leta in potem izpiši mrzle mesece, torej take, ki imajo temperaturo nižjo od povprečne.

```
static void Main(string[] args)
{
    double[] meseci = new double[12];
    double letnoPovp = 0;

    for (int i = 0; i < 12; i++)
    {
       Console.Write("Povprečna temperatura za " + (i + 1) + ". mesec : ");
       meseci[i] = double.Parse(Console.ReadLine());
       letnoPovp = letnoPovp + meseci[i];
    }

    letnoPovp = Math.Round(letnoPovp / 12, 2);
    Console.WriteLine("Povprečna letna temperatura : " + letnoPovp);
    Console.WriteLine("Mrzli meseci: ");

    for (int i = 0; i < 12; i++)
    {
        if (meseci[i] <= letnoPovp) Console.Write((i + 1) + ". mesec  ");
    }
} //main
```

Delitev znakov v stavku

Preberi poljuben stavek in izpiši, koliko znakov vsebuje, koliko je v njem samoglasnikov, koliko števk in koliko ostalih znakov

```
static void Main(string[] args)
{
    string samogl = "AEIOU";

    int stSam = 0, stStevk = 0;
    string stavek;

    Console.Write("Vnesi poljuben stavek: ");
```

```
    stavek = Console.ReadLine();
    Console.WriteLine();

    for (int i = 0; i < stavek.Length; i++)
    {
       char znak = stavek[i];
       if (samogl.IndexOf(znak) > -1) // znak je samoglasnik
          stSam = stSam + 1;
       if ('0' <= znak && znak <= '9')
          stStevk = stStevk + 1;
    }
    Console.WriteLine("\nŠtevilo vseh znakov v stavku : " + stavek.Length);
    Console.WriteLine("Število samoglasnikov        : " + stSam);
    Console.WriteLine("Število cifer                : " + stStevk);
    Console.WriteLine("Število ostalih znakov       : " +
         (stavek.Length - stSam - stStevk));
} //main
```

Zbiramo sličice

Začeli smo zbirati sličice. V album moramo nalepiti 250 sličic. Zanima nas, koliko sličic bomo morali kupiti, da bomo napolnili album. Seveda ob nakupu ne vemo, katero sličico bomo dobili. Ker bi to radi vedeli še preden se bomo zares spustili po nakupih, bi radi polnjenje albuma simulirali s pomočjo računalniškega programa. In če bomo to simulacijo ponovili dovolj mnogokrat, bomo že dobili občutek o številu potrebnih nakupov. Pri tem seveda naredimo nekaj predpostavk:

- hkrati vedno kupimo le eno sličico.
- Vse sličice so enako pogoste. Torej je verjetnost, da bomo kupili i-to sličico ravno 1/250.
- Podvojenih sličic ne menjamo.

Poglejmo, kako bi sestavili program.
Naš album bo tabela. Če bo vrednost ustreznega elementa *false*, to pomeni da sličice še nimamo. Da nam ne bo po vsakem nakupu treba prečesati vsega albuma, da bi ugotovili, ali kakšna sličica še manjka, bomo vodili evidenco o tem, koliko sličic v albumu še manjka. V zanki, ki se bo odvijala toliko časa, dokler število manjkajočih sličic ne bo padlo na nič, bomo kupili sličico (naključno generirali število med 0 in 249). Če je še nimamo, bomo zmanjšali število manjkajočih sličic in si v albumu označili, da sličico imamo.

```
static void Main(string[] args)
{
    int st_slicicvalbumu = 250;
    int st_zapolnjenih = 0;
    int st_kupljenihslic = 0;
    // boben nak. stevili
    Random bob = new Random();
    // album
    bool[] album = new bool[st_slicicvalbumu];
    // polnimo album
    while (st_zapolnjenih < st_slicicvalbumu)
    {
        st_kupljenihslic = st_kupljenihslic + 1;
        int st_slikic= bob.Next(st_slicicvalbumu);
        // jo damo v album
        if (!album[st_slikic])
        {
            album[st_slikic] = true;
            st_zapolnjenih = st_zapolnjenih + 1;
        }
    }
```

```
    Console.WriteLine("Da smo napolnili album, smo kupili "+
                          + st_kupljenihslic + " slikic.");
}
```

Program prevedemo in poženemo:

```
Da smo napolnili album, smo kupili 2265 slikic.
```

Opis programa:
V uvodu definiramo tri spremenljivke s katerimi nadzorujemo stanje album. V 9. vrstici začnemo polniti album, za to uporabili zanko *while*. Ob vsakem vstopu v zanko kupimo sličico , ki ji dodelimo naključno mesto v albumu (naključno generirali število med 0 in 249). Preverimo če že imamo to mesto zapolnjeno (*!album[st_slikic]*). Če ne, jo vstavimo v album in povečamo število zapolnjenih mest v albumu. To ponavljamo toliko časa, da je pogoj v zanki resničen. Na koncu še izpišemo, koliko sličic smo morali kupiti, da smo napolnili album.

Najdaljša beseda v nizu

Napišimo program, ki bo našel in izpisal najdaljšo besedo prebranega niza. Predpostavimo, da so besede ločene z enim presledkom.

Da bomo iz niza izluščili besede, si bomo pomagali z metodo Split, ki jo najdemo v razredu string. Metoda vrne tabelo nizov, v kateri vsak element predstavlja podniz niza, nad katerim smo metodo uporabili. V jeziku C# niz razmejimo na besede z razmejitvenim znakom oziroma znaki, ki jih ločimo z vejico (npr. Split('/')). Posamezen razmejitveni znak pišemo v enojnih narekovajih ('). V javi za razmejitev niza ne uporabljamo razmejitvenih znakov, temveč uporabimo razmejitveni niz (npr. split("/")), ki ga pišemo v navednicah ("). Metodo Split kot vedno v javi pišemo z malo začetnico.

Primer:

Denimo, da imamo

```
string stavek = "a/b/c.";
```

Če uporabimo

```
string[] tab = stavek.Split('/');
```

smo s tem ustvarili novo tabelo velikosti 3. V tabeli so shranjeni podnizi niza stavek, kot jih loči razmejitveni znak /. Prvi element tabele tab je niz "a", drugi element je niz "b" in tretji element niz "c.".

```
C1:  public static void Main(string[] args)
C2:  {
C3:      // Vnos niza
C4:      Console.Write("Vnesi niz: ");
C5:      string niz = Console.ReadLine();
C6:
C7:      // Pretvorba niza v tabelo nizov
C8:      string[] tab = niz.Split(' ');
C9:
C10:     // Iskanje najdaljše besede
C11:     string pomozni = ""; // Najdaljša beseda (oz. podniz)
C12:     for (int i = 0; i < tab.Length; i++)
C13:     {
C14:         if (pomozni.Length < tab[i].Length)
```

```
C15:          {
C16:              pomozni = tab[i];
C17:          }
C18:      }
C19:
C20:      // Izpis najdaljše besede
C21:      Console.WriteLine("Najdaljsa beseda: " + pomozni);
C22:  }
```

Zapis na zaslonu:

```
Vnesi niz: Lep poletni dan.
Najdaljsa beseda: poletni
```

Razlaga

Najprej določimo niz niz (C5). Nato s pomočjo klica metode Split() določimo tabelo tab (C8). V metodi za razmejitveni znak uporabimo presledek (' '). Če je med besedami niza niz več presledkov, se vsi presledki obravnavajo kot razmejitveni znaki. Nato določimo pomožni niz pomozni, ki predstavlja trenutno najdaljšo besedo (C11). Z zanko se sprehodimo po tabeli tab in poiščemo najdaljšo besedo. V vrstici C21 izpišemo najdaljšo besedo niza niz oziroma tabele tab.
Če za spremenljivko niz določimo prazen niz (v konzoli pri vnosu niza stisnemo le tipko Enter), se program izvede in za najdaljši niz izpiše prazen niz. Tabela tab je ostala prazna, zato je niz v spremenljivki pomozni ostal "".

Uredi elemente po velikosti

Sestavimo program, ki bo uredil vrednosti v tabeli celih števil po naslednjem postopku: Poiščimo najmanjši element in ga zamenjajmo s prvim. Nato poiščimo najmanjši element od drugega dalje in ga zamenjajmo z drugim, itn. Tabelo velikosti n preberemo in jo po urejanju izpišemo na ekran.

Sestavimo program po korakih:
Najprej deklariramo spremenljivko n, katere prebrano vrednost uporabimo za velikost tabele.

```
Console.Write("Velikost tabele: ");
int n = int.Parse(Console.ReadLine());
```

Nato deklariramo tabelo velikosti n.

```
int[] tab = new int[n];
```

Z zanko, ki se bo izvedla n-krat, napolnimo tabelo s števili, dobljenimi pri branju iz konzole.

```
Console.WriteLine();
for (int i = 0; i < n; i++)
{
    Console.Write("Vnesi " + (i + 1) + ". element: ");
    tab[i] = int.Parse(Console.ReadLine());
}
```

Nato naredimo zanko, ki se izvede (n-1)-krat. Ko nam bo ostal le zadnji element, bo že urejen. V zanki:

- Deklariramo pomožni spremenljivki. V prvi spremenljivki hranimo kandidata za najmanjši element med tistimi z indeksi od indeksa določenega z zanko do konca tabele. V drugi spremenljivki pa hranimo indeks kandidata, shranjenega v prvi spremenljivki.
- Naredimo zanko for, ki se sprehodi po tabeli od elementa, katerega indeks je določen s prvo zanko, do zadnjega elementa tabele.
    - Če najdemo manjši element, si zapomnimo njegovo vrednost in indeks.

- Izvedemo zamenjavo med trenutnim elementom in najdenim najmanjšim elementom.

```
for (int i = 0; i < n - 1; i++)
{
    int min = tab[i]; // Kandidat za najmanjši element od indeksa i
    // do konca tabele
    int indeks = i; // Indeks najmanjšega kandidata
    for (int k = i + 1; k < n; k++)
    {
        if (min > tab[k])
        { // Poiščimo najmanjši element v tem delu
            min = tab[k];
            indeks = k;
        }
    }
    // Zamenjava elementov
    int t = tab[i];
    tab[i] = tab[indeks];
    tab[indeks] = t;
}
```

Na koncu z zanko for, ki se sprehodi po urejeni tabeli tab, izpišemo elemente in jih pri izpisu ločimo s presledkom.

```
Console.WriteLine();
for (int i = 0; i < n; i++)
{
    Console.Write(tab[i] + " ");
}
// Prehod v novo vrstico
Console.WriteLine();
```

Poglejmo še zapis celotnega programa.

```
public static void Main(string[] args)
{
    // Vnos vrelikosti tabele
    Console.Write("Velikost tabele: ");
    int n = int.Parse(Console.ReadLine());

    // Deklaracija tabele
    int[] tab = new int[n];

    // Napolnitev tabele
    Console.WriteLine();
    for (int i = 0; i < n; i++)
    {
        Console.Write("Vnesi " + (i + 1) + ". element: ");
        tab[i] = int.Parse(Console.ReadLine());
    }

    // Uredimo tabelo
    for (int i = 0; i < n - 1; i++)
    {
        int min = tab[i]; // Kandidat za najmanjši element od indeksa i
        // do konca tabele
        int indeks = i; // Indeks najmanjšega kandidata
        for (int k = i + 1; k < n; k++)
        {// Poiščimo najmanjši element
```

```
            // v tem delu
            if (min > tab[k])
            {
                min = tab[k];
                indeks = k;
            }
        }
        // Zamenjava elementov
        int t = tab[i];
        tab[i] = tab[indeks];
        tab[indeks] = t;
    }

    // Izpis urejene tabele
    Console.WriteLine();
    for (int i = 0; i < n; i++)
    {
        Console.Write(tab[i] + " ");
    }
    // Prehod v novo vrstico
    Console.WriteLine();
}
```

Zapis na zaslonu:

```
Velikost tabele: 5

Vnesi 1. element: 6
Vnesi 2. element: 7
Vnesi 3. element: 3
Vnesi 4. element: 9
Vnesi 5. element: 1

1 3 6 7 9
```

Vsota elementov

Predpostavimo, da imamo tabelo z naslednjimi elementi: 2, 5, 7, 1, 6, 10, 3, 8, 0 in 11. Napišimo program, ki bo določil in izpisal vsoto vseh elementov.

```
1:  public static void Main(string[] args)
2:  {
3:      // Deklariramo tabelo in jo napolnimo z elementi
4:      int[] tab = { 2, 5, 7, 1, 6, 10, 3, 8, 0, 11 };
5:
6:      // Vsota elementov
7:      int vsota = 0;
8:
9:      // Ugotovitev vsote elementov tabele
10:     for (int i = 0; i < tab.Length; i++)
11:      {
12:       int element = tab[i];
13:          vsota = vsota + element;
14:      }
15:
16:      // Izpis vsote elementov tabele
17:      Console.WriteLine("Vsota elementov je " + vsota + ".");
18:  }
```

Zapis na zaslonu:

```
Vsota elementov je 53.
```

Razlaga. Najprej deklarirajmo tabelo tab in jo napolnimo z elementi. Zatem deklariramo spremenljivko vsota in ji priredimo začetno vrednost 0.
V vrstici 9 uporabimo zanko for, ki ponavlja stavka v vrsticah 12 in 13 toliko časa, niso uporabljeni vsi elementi tabele tab. V vrstici 13 seštevamo elemente tabele in njihovo vsoto hranimo v spremenljivki vsota. Po koncu zanke izpišemo skupno vsoto vseh elementov (17).

Zanka foreach

Zanka foreach je zanka, v kateri se ponavlja množica stavkov za vsak element neke tabele ali množice objektov. Zanke ne moremo uporabiti za spremembo elementov tabele, po kateri se sprehajamo, oz. za spremembo objektov. Na začetku foreach zanke je deklarirana spremenljivka poljubnega tipa, ki avtomatično pridobi vrednost posameznega elementa neke tabele, zaradi česar ima ta oblika zanke prednost pred klasičnim for stavkom za obdelavo neke tabele. Uporabimo jo lahko le za obdelavo cele tabele, ne pa tudi za obdelavo le njenega dela. Iteracija poteka vedno od indeksa 0 do indeksa Length -1, iteracija v obratni smeri pa NI možna.

Splošna oblika foreach zanke:

```
foreach ( tip spremenljivka in tabela)
{
  ..stavki..  //poljuben stavek ali več stavkov
}
```

Primer:

```
string znaki = "abcdefg";

//foreach zanka za dostop do vseh znakov v stringu
string znakiSPresledki = "";

foreach (char znak in znaki)
    znakiSPresledki += znak + " ";
//znakiSPresledki dobi vrednost "a b c d e f g "
```

Še en primer:

```
//deklaracija in inicializija enodimenzionalne tabele celih števil
int[] tab = { 0,1,2,3,4,5,6,7,8,9,10 };

Console.WriteLine("\nTabela tab vsebuje "+tab.Length+" elementov. \nTa
tabela je "+tab.Rank+". dimenzionalna");

Console.WriteLine(vsebina tabele: ");

foreach(int x in  tab)
    Console.Write(x+", "); //Izpis vsebine tabele
```

Manjkajoča števila

Generirajmo tabelo 50 naključnih števil med 0 in 100 (obe meji štejemo zraven). S pomočjo zanke foreach ugotovimo in izpišimo, katerih števil med 0 in 100 ni v tej tabeli. Za kontrolo naredimo izpis elementov tabele.

Sestavimo program po korakih:
Najprej deklariramo dve konstanti. Prvo uporabimo za velikost tabele, drugo pa za zgornjo mejo naključnih števil.

```
const int vel = 50; // Velikost tabele
const int mejaStevil = 100; // Zgornja meja naključnih števil
```

Nato deklariramo tabelo, ki bo ima velikost vel.

```
int[] tab = new int[vel];
```

Ustvarimo še generator naključnih števil.

```
    Random nak = new Random(); // Generator naključnih števil
```

Z zanko, ki se izvede vel-krat, napolnimo tabelo z naključnimi števili med 0 in 100 (obe meji štejemo zraven):

```
for (int i = 0; i < vel; i++)
{
    tab[i] = nak.Next(mejaStevil + 1); // Določitev naključnega števila
}
```

Nato z zanko for, ki se sprehodi po elementih tabele tab, izpišemo elemente tabele. Elemente tabele ločimo s presledkom.

```
Console.WriteLine("Izpis vsebine tabele nakljucnih stevil: ");
// Izpis elementov tabele
for (int i = 0; i < tab.Length; i++)
{
    int el = tab[i];
    Console.Write(el + " "); // Izpis elementa, ločenega s presledkom
}
```

Nato še enkrat naredimo zanko, ki se izvede 101-krat. V njej

- Ustvarimo logično spremenljivko za iskanje števila. Začetna vrednost spremenljivke je false.
- Naredimo zanko foreach, ki se sprehodi po elementih tabele tab. V zanki
  - Če število najdemo
    - vrednost logične spremenljivke nastavimo na true in
    - s stavkom break prekinemo iskanje.
- Če števila ne najdemo, ga izpišemo. Izpisana števila ločimo s presledkom.

```
Console.WriteLine("\n\nStevila med 0 in " + mejaStevil +
                                  ", ki niso v tej tabeli, so: ");
// Iskanje števil, ki se ne nahajajo v tabeli
for (int i = 0; i < mejaStevil + 1; i++){
    bool nasli = false; // Spremenljivka za iskanje števil
    foreach (int el in tab)
    {
      if(el == i)
      {
```

```
            nasli = true; // Število smo našli
            break; // Prekinemo iskanje števila
        }
    }
    // Ali smo število našli, nam pove spremenljivka nasli
   if (!nasli) Console.Write(i + " ");
}
```

Na koncu gremo še v novo vrstico.

```
Console.WriteLine();
```

Poglejmo sedaj še zapis celotnega programa.

```
public static void Main(string[] args)
{
    const int vel = 50; // Velikost tabele
    const int mejaStevil = 100; // Zgornja meja naključnih števil
    int[] tab = new int[vel]; // Dekleracija tabele
    Random nak = new Random(); // Generator naključnih števil

    // Polnjenje tabele
    for (int i = 0; i < vel; i++)
    {
        tab[i] = nak.Next(mejaStevil + 1); // Določitev naključnega števila
    }

    Console.WriteLine("Izpis vsebine tabele nakljucnih stevil: ");
    // Izpis elementov tabele
    for (int i = 0; i < tab.Length; i++)
    {
        int el = tab[i];
        Console.Write(el + " "); // Izpis elementa, ločenega s presledkom
    }

    Console.WriteLine("\n\nStevila med 0 in " + mejaStevil +
                ", ki niso v tej tabeli, so: ");
    // Iskanje števil, ki se ne nahajajo v tabeli
    for (int i = 0; i < mejaStevil + 1; i++)
    {
        bool nasli = false; // Spremenljivka za iskanje števil
        for (int j = 0; j < tab.Length; j++)
        {
            int el = tab[j];
            if (el == i)
            {
                nasli = true; // Število smo našli
                break; // Prekinemo iskanje števila
            }
        }
        // Ali smo število našli, nam pove spremenljivka našli
        if (!nasli) Console.Write(i + " ");
    }

    // Nova vrstica
    Console.WriteLine();
}
```

V napisanem programu smo uporabili algoritem, katerega časovna zahtevnost je O (velikost tabele x število vseh števil). Če pa si lahko "privoščimo" uporabo dodatne tabele take velikosti, kot je vseh možnih števil (v našem primeru 101), lahko razvijemo boljši (hitrejši) algoritem.

Za izboljšavo napisanega programa bomo uporabili pomožno tabelo, ki bo nadomestila tisti del "starega" programa, kjer smo uporabili gnezdeno zanko. V tabeli bomo vsem elementom, katerih indeksi so enaki izbranim številom, nastavili vrednosti na *true*. Po nastavitvi vrednosti bomo poiskali vse tiste elemente, ki so ohranili vrednost false. Kadar bomo našli tak element, bomo izpisali njegov indeks. Le ta bo neizbrano število.

Najprej iz programa Stevila.cs uporabimo vrstice

```
const int vel = 50; // Velikost tabele
const int mejaStevil = 100; // Zgornja meja naključnih števil
int[] tab = new int[vel]; // Dekleracija tabele
Random nak = new Random(); // Generator naključnih števil

// Polnjenje tabele
for (int i = 0; i < vel; i++)
{
    tab[i] = nak.Next(mejaStevil + 1); // Določitev naključnega števila
}

Console.WriteLine("Izpis vsebine tabele nakljucnih stevil: ");
// Izpis elementov tabele
for (int i = 0; i < tab.Length; i++)
{
    int el = tab[i];
    Console.Write(el + " "); // Izpis elementa, ločenega s presledkom
}
```

Določimo tabelo logičnih vrednosti, katere velikost je mejaStevil + 1 (vsa števila med 0 in 100, vključno z mejama).

```
bool[] pom = new bool[mejaStevil + 1]; // Pomožna tabela
```

Ob deklaraciji se vsi elementi tabele avtomatsko postavijo na false. S tem smo predpostavili, da podatek i ni v tabeli tab.

Nato naredimo zanko foreach, s katero v tabeli pom na true nastavimo tiste elemente, ki imajo indekse, ki so v tabeli tab.

```
// Označimo elemente, ki so v tabeli tab
foreach (int el in tab)
{
    pom[el] = true; // Vrednost elementa nastavimo na true
}
```

Sedaj moramo poskrbeti le še za izpis.
Zato naredimo zanko for, ki se sprehodi po tabeli pom.
Če je vrednost trenutnega elementa false, izpišemo njegov indeks. Indekse ločimo s presledkom.

```
Console.WriteLine("\n\nStevila med 0 in " + mejaStevil +
                  ", ki niso v tej tabeli, so: ");
// Iskanje števil, ki se ne nahajajo v tabeli
for (int i = 0; i < pom.Length; i++)
{
    if (pom[i] == false)
        Console.Write(i + " ");
```

```
}
// Nova vrstica
Console.WriteLine();
```

Poglejmo sedaj še zapis izboljšanega programa.

```
public static void Main(string[] args)
{
    const int vel = 50; // Velikost tabele
    const int mejaStevil = 100; // Zgornja meja naključnih števil
    int[] tab = new int[vel]; // Dekleracija tabele
    Random nak = new Random(); // Generator naključnih števil

    // Polnjenje tabele
    for (int i = 0; i < vel; i++)
    {
        tab[i] = nak.Next(mejaStevil + 1); // Določitev naključnega števila
    }

    Console.WriteLine("Izpis vsebine tabele nakljucnih stevil: ");
    // Izpis elementov tabele tab
    foreach (int el in tab)
    {
        Console.Write(el + " "); // Izpis elementa, ločenega s presledkom
    }

    bool[] pom = new bool[mejaStevil + 1]; // Pomožna tabela

    // Označimo elemente, ki so v tabeli tab
    foreach (int el in tab)
    {
        pom[el] = true; // Vrednost elementa nastavimo na true
    }

    Console.WriteLine("\n\nStevila med 0 in " + mejaStevil +
     ", ki niso v tej tabeli, so: ");
    // Iskanje števil, ki se ne nahajajo v tabeli
    for (int i = 0; i < pom.Length; i++)
    {
        if (pom[i] == false)
            Console.Write(i + " ");
    }
    // Nova vrstica
    Console.WriteLine();
}
```

Zapis na zaslonu:

```
Izpis vsebine tabele nakljucnih stevil:
2 43 5 85 17 32 97 72 72 23 18 95 4 86 42 51 1 9 14 57 79 99 37 85 72 63 17 67 43 92 47 57
 9 79 21 77 8 62 6 36 18 75 50 6 67 81 24 54 11 72

Stevila med 0 in 100, ki niso v tej tabeli, so:
0 3 7 10 12 13 15 16 19 20 22 25 26 27 28 29 30 31 33 34 35 38 39 40 41 44 45 46 48 49 52
53 55 56 58 59 60 61 64 65 66 68 69 70 71 73 74 76 78 80 82 83 84 87 88 89 90 91 93 94 96
98 100
```

Naloge za utrjevanje znanja iz tabel

- Napišite program, ki prebere tri cela števila n, a in b ter tabelo velikosti n napolni z naključnimi števili med a in b. Tabelo naj potem program izpiše tako, da v vrstici izpiše po 5 števil.

- Sestavite program, ki bo prebral podatke v tabelo celih števil, nato pa preveril, ali tabela predstavlja permutacijo števil od 1 do n, kjer je n dolžina tabele.
  Namig: Tabela dolžine n predstavlja permutacijo, če v njej vsako naravno število od 1 do n nastopa natanko enkrat.

- Dana je tabela [1, 6, 4 ,2, 8, 3, 7]. Uredite jo po naslednjem postopku: Poiščite največji element in ga zamenjajte z zadnjim. Nato poiščite največji element od prvega do predzadnjega in ga zamenjajte s predzadnjim …

- Generirajte 1000 naključnih števil med 0 in 10 in preštejete, kolikokrat se vsak element pojavi.

- V tabelo dolžine n zapišite naključne male črke angleške abecede. Sestavite novo tabelo, v kateri se vsak element prvotne tabele ponovi natanko dvakrat. Novo tabelo izpišite na zaslon.
  Primer:
  Če je prvotna tabela ['a', 'n', 'c'], je nova tabela ['a', 'a', 'a', 'n', 'n', 'n', 'c', 'c','c'].

- V tabelo dolžine 30 zapišite naključna števila od 0 do 20 in jih izpišite kot množico: vsako samo enkrat.

- Sestavite program, ki napolni tabelo velikost n z naključnimi nenegativnimi celimi števili. Prepišite te elemente v novo tabelo tako, da je i-ti element razlika med vsoto vseh elementov prvotne tabele in i-tim elementom prvotne tabele. Izpišite obe tabeli tako, da v vrsti izpišete po 10 elementov.

Dvodimenzionalne in večdimenzionalne tabele

Dvodimenzionalne tabele vsebujejo vrstice in stolpce.

Splošna deklaracija dvodimenzionalne tabele:

```
Podatkovni_tip[ , ] ime_tabele = new Podatkovni_tip [vrstic, stolpcev];
```

Tudi dvodimenzionalno tabelo lahko deklariramo na tri načine:

- Najprej deklariramo tabelo (npr z imenom **tabela)** , kasneje pa ji določimo še velikost:

```
int[,] tabela;  //deklaracija tabelarične spremenljivke
tabela = new int[100 , 200]; //dvodimenzionalna tabela 100 vrstic in
200 stolpcev
```

- Ob deklaraciji tabelarične spremenljivke z operatorjem **new** takoj zasežemo pomnilnik:

```
int[ , ] tabela = new int[100 , 200];
```

- Tabelo najprej deklariramo, z operatorjem **new** zasežemo pomnilnik, nato pa jo še inicializiramo

```
int[ , ] tabela = new int[ 2, 3]  { { 1, 1 ,1 },{2, 2, 2 }};
```

Tudi v prvih dveh primerih se spremenljivke, samodejno inicializirajo (v našem primeru ima vseh 100 x 200 (to je 20.000) elementov tabele vrednost 0.

Primer:

Deklariraj dvodimenzionalno tabelo 10 x 10 celih števil in jo inicializiraj tako, da bodo po diagonali same enice, nad diagonalo same dvojke, pod diagonalo pa ničle.

```
int[,] tabela=new int[10,10];  //deklaracija tabelarične spremenljivke
//ob  deklaraciji  se  je  avtomatični  izvedla  še  inicializacija:  vse
tabelarične spremenljivke so //dobile vrednost 0

for (int i=0;i<10;i++)
{
    for (int j = 0; j < 10; j++)
    {
        if (i == j)
             tabela[i, j] = 1;
        else if (i<j)
             tabela[i, j] = 2;
    }
}
//tabelo še izpišimo v primerni obliki
for (int i = 0; i < 10; i++)
{
     for (int j = 0; j < 10; j++)
         Console.Write(tabela[i, j] + "  ");
     Console.WriteLine();
}
```

Poštevanka

Kreiraj dvodimenzionalno tabelo 10 x 10 celih števil. V posamezni vrstici naj bo poštevanka števil med 1 in 10. Tabelo na primeren način tudi izpiši.

```
int [,] tabela=new int[10,10];
for (int i=0;i<10;i++)
    for (int j=0;j<10;j++)
        tabela[i,j]=(i+1)*(j+1);

//Še izpis tabele
for (int i=0;i<10;i++)
{
    for (int j=0;j<10;j++)
        Console.Write(tabela[i,j] + "\t");
    Console.WriteLine();
}
```

Največja vrednost v tabeli

Kreiraj naključne elemente kvadratne matrike (4x4) celih števil med 0 in 10 in nato izpiši največjo vrednost v tabeli. Ugotovi in izpiši, na katerih mestih se ta največji element pojavi v tabeli (indeksi!).

```
int[,] mat = new int[4, 4];
int i, j, max = 0;

//generator naključnih vrednosti
Random Nakljucno = new Random();
for (i = 0; i < 4; i++)
{
    for (j = 0; j < 4; j++)
    {
         mat[i, j] = Nakljucno.Next(10);
         if ((i == 0) && (j == 0)) max = mat[i, j]; //iskanje najvecjega
         if (max < mat[i, j]) max = mat[i, j];
    }
}

Console.WriteLine("TABELA 4 x 4\n");
for (i = 0; i < 4; i++)
{ //izpis matrike
    for (j = 0; j < 4; j++)
    Console.Write(mat[i, j] + " ");
    Console.WriteLine();
}
Console.WriteLine("\nNajvecji element: " + max + "\n");
Console.WriteLine("Indeksi najvecjega elementa: ");
for (i = 0; i < 4; i++)  //izpis pozicij/indeksov najvecjega elementa
    for (j = 0; j < 4; j++)
        if (mat[i, j] == max) Console.WriteLine("[" + i + "," +  j + "]",
i, j);
```

Naloge

- Ustvari naključno tabelo 100 celih števil z vrednostmi med 10 in 100 in izpiši le tiste člene, ki so večji od zadnjega elementa v tej tabeli. Program dopolni tako, da ustvariš naključno dolgo tabelo (a ne krajšo od 10 in ne daljšo od 100 elementov).

- Zgeneriraj tabelo naključnih naravnih števil in izpiši vse elemente, ki so deljivi s številom, ki je na sredini tabele. Indeks elementa na sredini tabele je definiran kot (dolzina tabele) / 2.

- Sestavi program, ki bo uporabniku zastavil 10 računov za množenje celih števil. Eden izmed faktorjev naj bo naključno trimestno celo število, drugi izmed faktorjev pa mora biti naključno enomestno število. Števila in uporabnikove rezultate shranjuj v ustrezno tabelo. Tabelo NA KONCU obdelaj (preveri rezultate) in izpiši število pravilnih in število nepravilnih odgovorov.

- Napiši program, ki prebere tri cela števila: dimenzija, spMeja in zgMeja. Program naj nato tabelo velikosti dimenzija napolni z naključnimi števili med spMeja in zgMeja.

- Beri besede in jih shranjuj v tabelo dimenzije do 100 elementov. Ugotovi in izpiši najdaljšo besedo v tej tabeli. Napiši funkcijo, ki dobi za parameter to tabelo in poljubno celo število N in ki vrne podatek o tem, koliko besed v tej tabeli vsebuje več kot N znakov!

- Napiši program, ki bo izpisal dvodimenzionalno tabelo naključnih celih števil med 0 in 100 (dimenziji izbere uporabnik programa), poiskal njen največji element ter izračunal, za koliko se največji element razlikuje od povprečne vrednosti vseh elementov.

- Napiši program, ki prešteje število sodih in lihih elementov v naključno generirani tabeli celih števil. Nato iz začetne tabele sestavi dve tabeli. V prvi so samo soda števila, v drugi pa samo liha.

- Sestavi tabelo naključnih celih števil med 1 in 100. Prepiši jih v novo tabelo tako, da bodo v novi tabeli najprej elementi, ki imajo v prvi tabeli indekse 0, 2, 4, ..., potem pa še elementi z lihimi indeksi. Primer: Če ime prvotna tabela elemente 2, 4, 23, 5, 45, 6, 8 so v novi tabeli elementi razporejeni kot 2, 23, 45, 8, 4, 5, 6. 3. Izpiši obe tabeli po 10 v vrsto. Za realizacijo naloge napiši metode: generiraj - ustvari tabelo, prelozi - preloži v novo tabelo na zahtevan način in izpisi - v vsaki vrsti se izpiše 10 elementov

- Dan je zelo dolg niz, sestavljen iz števk. Ugotovi, katerih števk je v nizu največ.

- Dana je dvodimenzionalna tabela 10 x 10 celih števil. Napiši funkcijo, ki dobi to tabelo za parameter in ki vrne vsoto vseh sodih števil iz te tabele. Za obdelavo tabele uporabi zanko foreach.

- Dana je enodimenzionalna tabela 100 znakov. Ugotovi, ali se v tej tabeli nahaja določen znak.

- Dana je premica y = 2x+3. Napiši program, ki bo najprej zgeneriral tabelo TOCKE 10 naključnih celih števil med -20 in + 20 (POZOR! – števila morajo biti različna) nato pa izpisal koordinate desetih točk, ki ležijo na dani premici, pri čemer boš za prve koordinate vzel vrednosti iz tabele TOCKE.

- Ustvari naključno tabelo 100 celih števil z vrednostmi med 50 in 100 in nato izpiši le tiste člene, ki so večji od zadnjega. Program dopolni še tako, da ustvariš naključno dolgo tabelo (a ne krajšo od 10 in ne daljšo od 1000 elementov).

- Napiši program, ki prebere določeno število nizov in jih izpiše v obratnem vrstnem redu. Podatek o tem, koliko nizov bo vnesenih, prebereš na začetku programa. Pomagaj si s tabelo.

- Najprej smiselno (na dveh mestih) dopolni program

```
public class TabelaImen
{
  public static void Main (string[] args)
  {
```

```
    string[] ________= {"Mojca", "Katja", "Gašper", "Gabriela", "Tine"};

    System.Console.WriteLine("Dolžina tabele je " + _____________ + ".");
    System.Console.WriteLine("Ime v tabeli je " + imena[3] + ".");
   }
}
```

Nato odgovori: Kaj se izpiše, ko program prevedemo in poženemo?

- Dana sta ukaza

```
string[] stavek = new string[10];
stavek[0] = "Konec se bliza";
```

Kateri ukaz moramo uporabiti, če želimo izpisati dolžino niza "Konec se bliza"? Obkroži vse pravilne odgovore!

```
a) System.Console.WriteLine(stavek.Length());
b) System.Console.WriteLine(stavek[0].Length);
c) System.Console.WriteLine(stavek.Length);
d) System.Console.WriteLine(stavek[0].Length());
```

- Napiši program, ki prebere dve tabeli celih števil. Izpiše naj tiste vrednosti, ki se pojavijo v drugi tabeli in ne v prvi! Vemo, da so vse vrednosti v obeh tabelah med seboj različne (torej se v isti tabeli število nikoli ne ponovi)

  Primeri:

  - Če so v prvi tabeli podatki 3, 4, 2, 1 in v drugi 3, 4, 6, 5, naj program izpiše podatka 3 in 4.
  - Če so v prvi tabeli podatki 3, 41, 2, 11 in v drugi 3, 11, 41, 2 naj program ne izpiše nič.
  - Če so v prvi tabeli podatki 3, 41, 2, 11 in v drugi 100, 17, 11, 13, 41, 2 naj program izpiše 100, 17 in 13.

- Kaj izpiše spodnji del programa:

```
string[] niz = {"Tantadruj"};
int x;
x = niz.Length;
System.Console.WriteLine(x);
```