ktu informatikos fakultetas 1922

T10. Teksto analizė ir redagavimas.

4 ak. val.

ktu informatikos fakultetas 1922

Temos klausimai

1. Tipas **`char`** ir jo metodai.
2. Klasė **`string`** ir jos metodai.
3. Klasė **`StringBuilder`** ir jos metodai.
4. Teksto analizė ir redagavimas.
5. Teksto apdorojimo eilutėmis algoritmas.
6. Failo redagavimas: nurodytos failo eilutės pašalinimas, naujos įterpimas, eilučių sukeitimas vietomis faile.

*Bazinis duomenų tipas **char***

ktu informatikos fakultetas 1922

char duomenų tipas 1

char – bazinis tipas.

Užimamos atminties kiekis - 16 bitų.

Naudojamos koduotės:

Unicode – Windows (65536 simboliai),

Visual Studio – UTF-16.

Numatytoji reikšmė - '\u0000'.

Minamali reikšmė - '\u0000'.

Maksimali reikšmė - '\uffff'.

Lietuviškų raidžių koduotė **Encoding.GetEncoding(1257).**

Unicode formatas – 2 baitai, įprastas ASCII – 1 baitas.

ktu informatikos fakultetas 1922

char duomenų tipas 2

- didžiosios lotynų kalbos raidės **‘A’ . . . ‘Z’** ;
- mažosios lotynų kalbos raidės **‘a’ . . . ‘z’** ;
- skaitmenys **‘0’ . . . ‘9’** ;
- kiti teksto simboliai **‘,’, ‘;’, ‘?’** ir t.t.
- valdantieji simboliai (toliau);
- simbolių kodai – sveikieji skaičiai nuo **0** iki **65535**.

ktu informatikos fakultetas 1922

char duomenų tipas 3

Simbolis	Paskirtis
`\n`	Nauja eilutė
`\t`	Horizontalioji tabuliacija
`\v`	Vertikalioji tabuliacija
`\b`	Grįžimas per simbolį atgal
`\r`	Grįžimas į eilutės pradžią
`\f`	Popieriaus patraukimas iki puslapio pabaigos
`\\`	Kairysis brūkšnys
`\'`	Apostrofas
`\"`	Kabutės
`\?`	Klaustukas
`\xhhh`	Simbolio ASCII kodas – nuo vieno iki trijų šešioliktainių skaitmenų. Pvz.: \x41 žymi didžiąją A raidę.

char duomenų tipas 4

Valdantieji simboliai valdo teksto spausdinimą. Dauguma iš jų savo funkcijos atlikimui pasislepia tekste – tampa nematomais.

Dažniausiai sutinkami nematomi teksto simboliai yra:

`'\n'` – naujos eilutės simbolis;

`eof` – failo pabaigos simbolis;

Tai C# kalbos žymėjimai.

char duomenų tipas 5

ktu informatikos fakultetas 1922

Skaitmenys

0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Raidės

A	65	N	78
B	66	O	79
C	67	P	80
D	68	Q	81
E	69	R	82
F	70	S	83
G	71	T	84
H	72	U	85
I	73	V	86
J	74	W	87
K	75	X	88
L	76	Y	89
M	77	Z	90

a	97	n	110
b	98	o	111
c	99	p	112
d	100	q	113
e	101	r	114
f	102	s	115
g	103	t	116
h	104	u	117
i	105	v	118
j	106	w	119
k	107	x	120
l	108	y	121
m	109	z	122

char duomenų tipas 6

Dažniausiai naudojami simboliai Unicode lentelėje surašyti 32 – 127 vietose.

Lietuviškos abėcėlės raidės surašytos:

ą – 261; Ą – 260; **č – 269; Č – 268;**

ę – 281; Ę – 280; **ė – 279; Ė – 278;**

į – 303; Į – 302; **š – 353; Š – 352;**

ų – 371; Ų – 370; **ū – 363; Ū – 362;**

ž – 382; Ž – 381.

ktu informatikos fakultetas 1922

char kintamųjų užrašymas

```
char simb1, simb2, simb3, simb4, simb5;
simb1 = 'X';          // simbolis
simb2 = '\x0058';     // šešioliktainis kodas
simb3 = (char)65;     // konvertuoti iš
simb4 = (char)126;    // sveiko skaičiaus
simb5 = '\u0033';     // Unikodas
char simb6 = '\"';    // valdymo simboliams
                      // priekyje rašomas \
```

char kintamųjų masyvas

```
char[] simboliai = new char[5];
simboliai[0] = 'L';
simboliai[1] = 'a';
simboliai[2] = 'b';
simboliai[3] = 'a';
simboliai[4] = 's';
foreach (char c in simboliai)
    Console.Write(c + " ");
Console.WriteLine();
for (int i = 0; i<5;i++)
    Console.Write(simboliai[i] + " ");
```

char tipo metodai 1

(char)Console.Read() – skaito simbolį;

Convert.ToChar(eilutė) – konvertuoja į simbolį;

Char.IsDigit(kintamasis) – tikrina, ar skaitmuo (atsakymas true; false);

Char.IsLetter(kintamasis) – tikrina, ar raidė (atsakymas true; false);

Char.IsLetterOrDigit(kintamasis) – tikrina, raidė skaitmuo (atsakymas true; false);

Char.IsSymbol(kintamasis) – tikrina, ar simbolis (atsakymas true; false);

char tipo metodai 2

Char.IsLower(kintamasis) - tikrina, ar mažoji raidė (atsakymas true; false);

Char.IsUpper(kintamasis) - tikrina, ar mažoji raidė (atsakymas true; false);

Char.IsPunctuation(kintamasis) - tikrina, ar skyriklis (atsakymas true; false);

Char.ToLower(kintamasis) – konvertuoja į mažąją raidę;

Char.ToUpper(kintamasis) – konvertuoja į didžiąją raidę.

char tipo metodų pavyzdžiai 1

```
Console.Write("Įveskite simbolį: ");
char sim = Convert.ToChar(Console.ReadLine());
Console.WriteLine();
Console.WriteLine("Ar skaičius?: {0}", Char.IsDigit(sim));
Console.WriteLine("Ar raidė?: {0}", Char.IsLetter(sim));
Console.WriteLine("Raidė ar skaičius?: {0}",
        Char.IsLetterOrDigit(sim));
Console.WriteLine("Ar mažoji raidė?: {0}", Char.IsLower(sim));
Console.WriteLine("Ar didžioji raidė?: {0}", Char.IsUpper(sim));
Console.WriteLine("Į mažąją: {0}", Char.ToLower(sim));
Console.WriteLine("Į didžiąją?: {0}", Char.ToUpper(sim));
Console.WriteLine("Ar skyriklis?: {0}",Char.IsPunctuation(sim));
Console.WriteLine("Ar simbolis?: {0}", Char.IsSymbol(sim));
```

char tipo metodų pavyzdžiai 2

```
Įveskite simbolį: Z

Ar skaičius?: False
Ar raidė?: True
Raidė ar skaičius?: True
Ar mažoji raidė?: False
Ar didžioji raidė?: True
Į mažąją: z
Į didžiąją?: Z
Ar skyriklis?: False
Ar simbolis?: False
Press any key to continue . . .
```

```
Įveskite simbolį: ;

Ar skaičius?: False
Ar raidė?: False
Raidė ar skaičius?: False
Ar mažoji raidė?: False
Ar didžioji raidė?: False
Į mažąją: ;
Į didžiąją?: ;
Ar skyriklis?: True
Ar simbolis?: False
Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Klasė `string` *ir jos metodai*

ktu informatikos fakultetas 1922

`string` duomenų tipas

`string` – išvestinis tipas.

Naudojamos koduotės:

Unicode – Windows (65536 simboliai),

Visual Studio – UTF-16.

Numatytoji reikšmė - **`null`**.

Lietuviškų raidžių koduotė **`Encoding.GetEncoding(1257).`**

string eilučių lyginimas

Visi simboliai simbolių eilutėje keičiami simbolių kodais.

Eilučių lyginimo operatoriai eilutes lygina nuosekliai pagal simbolių kodus eilutėse.

Eilutė didesnė, jeigu atitinkamas simbolis eilutėje didesnis.

Pvz.:

```
Namas
78 97 109 97 115
```

```
Naras
78 97 114 97 115
```

Verbatim eilutės

Naudojom

```
const string CFD = "..\\..\\Duomenys.txt";
```

Kad reiktų mažiau \

```
const string CFD = @"..\..\Duomenys.txt";
```

Antrąjį variantą galima skaldyti per keletą eilučių.

string klasės konstruktoriai 1

```
string eil = "..." - priskiriama eilutė;
string eil1 = eil  - priskiriamas string tipo kintamasis;
string eil2 = new string(char tipo masyvas) – įrašomas char
                          tipo masyvas;
string eil3 = new string(char tipo masyvas, prad indeksas,
              kiekis) – įrašoma char tipo masyvo dalis;
string eil4 = new string(simbolis, pasikartojimų kiekis) –
                        simbolis įrašomas norimą kartų kiekį.
```

string klasės konstruktoriai 2

```
char[] simboliai1 = { 'V', 'y', 'k', 's', 't', 'a', ' ',
                      'p', 'a', 's', 'k', 'a', 'i', 't', 'a'};
string eil = "Sveiki!";
string eil1 = eil;
string eil2 = new string(simboliai1);
string eil3 = new string(simboliai1, 7, 8);
string eil4 = new string('-', 15);
Console.WriteLine(eil);
Console.WriteLine(eil1);
Console.WriteLine(eil2);
Console.WriteLine(eil3);
Console.WriteLine(eil4);
```

```
Sveiki!
Sveiki!
Vyksta paskaita
paskaita
---------------
Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Eilučių sudėtis 1

- Sudėti (**+**) galima ne tik kelis **`string`** tipo objektus, bet ir **`string`** objektą su simboliu, tačiau rezultato eilutė visuomet turi būti nauja.
- Simbolių masyvas gali virsti eilute.
- Negalima simbolio tiesiogiai priskirti **`string`** tipo objektui.
- Simboliui galima priskirti vieną eilutės elementą.
- Eilutės elementai gali būti indeksuojami.

ktu informatikos fakultetas 1922

Eilučių sudėtis 2

```
char[] simboliai1 = {'p', 'a', 's', 'k', 'a', 'i', 't', 'a'};
string eil5;
eil5 = "" + simboliai1[0];
Console.WriteLine(eil5);
eil5 = "" + '5';
Console.WriteLine(eil5);
eil5 = eil + " " + eil1 + " " + eil2;

eil5 = '5'; // klaida
char simb6 = eil5[0]; //simboliui priskirtas eilutės elementas
```

Eilučių metodai 1

Eil.Length – eilutės ilgis;

Eil.CopyTo(pr_nr, char_masyvas, ind, simb_kiek) – Eil eilutės simbolius kopijuoja (konvertuoja) į char_masyvą; pr_nr – pradinio simbolio numeris, simb_kiek – simbolių kiekis, ind – pradžios indeksas masyve;

char_masyvas = Eil.ToCharArray(pr_nr, simb_kiek) – eilutės simbolius surašo (konvertuoja) į char_masyvą;

Eil1.Equals(Eil2) – lygina dvi eilutes (atsakymas true; false);

Eil1 == Eil2 - lygina dvi eilutes (atsakymas true; false). Galima lyginti == arba !=;

string.Equals(Eil1, Eil2) - lygina dvi eilutes (atsakymas true; false).

Eilučių metodai 2

Eil1.CompareTo(Eil2) – palygina eilutes. Atsakymas: 1 - Eil1>Eil2; 0 –lygios; -1 - Eil1<Eil2;

Eil1.StartsWith(Eil2) – tikrina, ar eilutė Eil1 prasideda eilute Eil2 (atsakymas true; false);

Eil1.EndsWith(Eil2) – tikrina, ar eilutė Eil1 pasibaigia eilute Eil2 (atsakymas true; false);

Eil1.IndexOf(simbolis) – randa simbolio pirmą vietą eilutėje, jei nėra – atsakymas -1;

Eil1.IndexOf(simbolis, pr) – randa simbolio pirmą vietą eilutėje, pradedant nuo simbolio, kurio numeris pr;

Eil1.IndexOf(simbolis, pr, kiekis) – randa simbolio pirmą vietą eilutėje, pradedant nuo simbolio, kurio numeris pr, ir tikrinant simbolių kiekį kiekis;

ktu informatikos fakultetas 1922

Eilučių metodai 3

Eil1.LastIndexOf(simbolis) – randa simbolio paskutinę vietą eilutėje, jei nėra – atsakymas -1;

Eil1.LastIndexOf(simbolis, gal) – randa simbolio paskutinę vietą eilutėje iki simbolio, kurio numeris gal;

Eil1.LastIndexOf(simbolis, gal, kiekis) – randa simbolio paskutinę vietą eilutėje iki simbolio, kurio numeris gal, ir tikrinant simbolių kiekį kiekis;

Eil1.IndexOf(Eil2) – randa eilutės Eil2 pirmą vietą eilutėje Eil1, jei nėra – atsakymas -1;

Eil1.IndexOf(Eil2, pr) – randa eilutės Eil2 pirmą vietą eilutėje Eil1, pradedant nuo simbolio, kurio numeris pr;

Eil1.IndexOf(Eil2, pr, kiekis) – randa eilutės Eil2 pirmą vietą eilutėje Eil1, pradedant nuo simbolio, kurio numeris pr, ir tikrinant simbolių kiekį kiekis;

Eilučių metodai 4

Eil1.LastIndexOf(Eil2) – randa eilutės Eil2 paskutinę vietą eilutėje Eil1, jei nėra – atsakymas -1;

Eil1.LastIndexOf(Eil2, gal) – randa eilutės Eil2 paskutinę vietą eilutėje Eil1 baigiant simboliu, kurio numeris gal;

Eil1.LastIndexOf(Eil2, pr, kiekis) – randa eilutės Eil2 paskutinę vietą eilutėje Eil1 baigiant simboliu, kurio numeris gal, ir tikrinant simbolių kiekį kiekis;

Eil1.IndexOfAny(char_mas) – randa simbolių masyvo bet kurio simbolio pirmą vietą eilutėje Eil1, jei nėra – atsakymas -1;

Eil1.IndexOfAny(char_mas, pr) – randa simbolių masyvo bet kurio simbolio pirmą vietą eilutėje Eil1, pradedant nuo simbolio, kurio numeris pr;

Eilučių metodai 5

Eil1.IndexOfAny(char_mas, pr, kiekis) – randa simbolių masyvo bet kurio simbolio pirmą vietą eilutėje Eil1, pradedant nuo simbolio, kurio numeris pr, ir tikrinant simbolių kiekį kiekis;

Eil1.LastIndexOf(char_mas) – randa simbolių masyvo bet kurio simbolio paskutinę vietą eilutėje Eil1,
jei nėra – atsakymas -1;

Eil1.LastIndexOf(char_mas, gal) – randa simbolių masyvo bet kurio simbolio paskutinę vietą eilutėje Eil1, baigiant simboliu, kurio numeris gal;

Eil1.LastIndexOf(char_mas, pr, kiekis) – randa simbolių masyvo bet kurio simbolio paskutinę vietą eilutėje Eil1, baigiant simboliu, kurio numeris gal, ir tikrinant simbolių kiekį kiekis;

Eilučių metodai 6

Eil1.Substring(pr) – išskiria eilutės dalį, pradedant simboliu, kurio numeris pr;

Eil1.Substring(pr, kiek) – išskiria eilutės dalį, kurios simbolių kiekis kiek, pradedant simboliu, kurio numeris pr;

Eil1.Replace(simb1, simb2) – eilutėje visus simbolius simb1 keičia simboliais simb2;

Eil1.Trim() – išmeta tarpus eilutės pradžioje ir pabaigoje;

Eil1.TrimStart() – išmeta tarpus eilutės pradžioje;

Eil1.TrimEndStart() – išmeta tarpus eilutės pabaigoje;

Eilučių metodai 7

Eil1.Trim(char_mas) – išmeta visus simbolių masyvo simbolius esančius eilutės Eil1 pradžioje ir pabaigoje;

Eil1.TrimStart(char_mas) – išmeta visus simbolių masyvo simbolius esančius eilutės Eil1 pradžioje;

Eil1.TrimEndStart(char_mas) – išmeta visus simbolių masyvo simbolius esančius eilutės Eil1 pabaigoje;

Eil1.ToUpper() – eilutės raides verčia didžiosiomis;

Eil1.ToLower() – eilutės raides verčia mažosiomis;

Eil1.Insert(pr, Eil2) – į eilutę, pradedant indeksu pr, įterpia eilutę Eil2. Suformuoja naują eilutę;

Eilučių metodai 8

Eil1.Split(char_mas, StringSplitOptions.RemoveEmptyEntries) – dalina eilutės Eil1 simbolius į grupes, char_mas – skyriklių masyvas. Dalinant išmeta tarpus;

Eil1.Split(eil_mas, StringSplitOptions.RemoveEmptyEntries) – dalina eilutės Eil1 simbolius į grupes, eil_mas – skyriklių eilučių masyvas. Dalinant išmeta tarpus;

Eil1.Remove(pr) – išmeta eilutės dalį, pradedant nurodytu indeksu. Formuoja naują eilutę;

Eil1.Remove(pr, kiekis) – išmeta eilutės dalį, kurios ilgis kiekis, pradedant nurodytu indeksu. Formuoja naują eilutę.

Eilučių metodų pavyzdžiai 1

```
string eil1 = "Vyksta paskaita";
char[] simboliai = new char[20];
char[] simboliai1 = new char[20];
Console.WriteLine("Eilutės eil1 ilgis " + eil1.Length);

Console.WriteLine("Eilutė: {0}", eil1);
Console.Write("Simbolių masyvas: ");
// kopijuoti eilutę į simbolių masyvą
eil1.CopyTo(7, simboliai, 0, 8);

for (int i = 0; i < simboliai.Length; ++i)
    Console.Write(simboliai[i]);
Console.WriteLine();
```

Eilučių metodų pavyzdžiai 2

```
// kopijuoti eilutę į simbolių masyvą
simboliai1 = eil1.ToCharArray(0, 6);
Console.Write("Simbolių masyvas1: ");

for (int i = 0; i < simboliai1.Length; ++i)
    Console.Write(simboliai1[i]);
Console.WriteLine();
```

```
Eilutės eil1 ilgis 15
Eilutė: Vyksta paskaita
Simbolių masyvas: paskaita
Simbolių masyvas1: Vyksta
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 3

```
string eil1 = "Pirmadienis";
string eil2 = "Antradienis";
string eil3 = "Trečiadienis";
string eil4 = "trečiadienis";
Console.WriteLine("eil1 = " + "\"" + eil1 + "\"\n" +
                  "eil2 = " + "\"" + eil2 + "\"\n" +
                  "eil3 = " + "\"" + eil3 + "\"\n" +
                  "eil4 = " + "\"" + eil4 + "\"\n");
if (eil1.Equals("Pirmadienis"))Console.WriteLine("eil1 lygi \"Pirmadienis\"");
else Console.WriteLine("eil1 nelygi \"Pirmadienis\"");
```

Eilučių metodų pavyzdžiai 4

```
if (eil1 == "Pirmadienis")Console.WriteLine("eil1 lygi \"Pirmadienis\"");
Else Console.WriteLine("eil1 nelygi \"Pirmadienis!\"");

if (string.Equals(eil3, eil4))Console.WriteLine("eil3 ir eil4 lygios");
else Console.WriteLine("eil3 ir eil4 nelygios");
```

```
eil1 = "Pirmadienis"
eil2 = "Antradienis"
eil3 = "Trečiadienis"
eil4 = "trečiadienis"

eil1 lygi "Pirmadienis"
eil1 lygi "Pirmadienis"
eil3 ir eil4 nelygios
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 5

```
string eil1 = "Pirmadienis"; string eil2 = "Antradienis";
string eil3 = "Trečiadienis"; string eil4 = "trečiadienis";

Console.WriteLine("eil1 = " + "\"" + eil1 + "\"\n" +
                  "eil2 = " + "\"" + eil2 + "\"\n" +
                  "eil3 = " + "\"" + eil3 + "\"\n" +
                  "eil4 = " + "\"" + eil4 + "\"\n");
Console.WriteLine("\neil1.CompareTo(eil2) yra " +
        eil1.CompareTo(eil2) + "\n" +
        "eil2.CompareTo(eil1) yra " +
        eil2.CompareTo(eil1) + "\n" +
        "eil1.CompareTo(eil1) yra " +
        eil1.CompareTo(eil1) + "\n" +
        "eil3.CompareTo(eil4) yra " +
        eil3.CompareTo(eil4) + "\n" +
        "eil4.CompareTo(eil3) yra " +
        eil4.CompareTo(eil3) + "\n");
```

```
eil1 = "Pirmadienis"
eil2 = "Antradienis"
eil3 = "Trečiadienis"
eil4 = "trečiadienis"

eil1.CompareTo(eil2) yra 1
eil2.CompareTo(eil1) yra -1
eil1.CompareTo(eil1) yra 0
eil3.CompareTo(eil4) yra 1
eil4.CompareTo(eil3) yra -1

Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 6

```
string[] eilutės = { "pirmas", "antras", "ketvirtas", "pirmadienis" };
for (int i = 0; i < eilutės.Length; ++i)
     if (eilutės[i].StartsWith("pir"))
         Console.WriteLine("\"" + eilutės[i] + "\"" +
                           " prasideda su \"pir\"");
     Console.WriteLine();
for (int i = 0; i < eilutės.Length; ++i)
     if (eilutės[i].EndsWith("is"))
         Console.WriteLine("\"" + eilutės[i] + "\"" +
                           " pasibaigia su \"is\"");
```

```
"pirmas" prasideda su "pir"
"pirmadienis" prasideda su "pir"

"pirmadienis" pasibaigia su "is"
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 7

```
string raidės = "aąbcdeęė kkk fcdc bghiį 123 jklmną  kloiiii";
char[] paieška = { 'ą', 'd', 'k', 's', '1' };
Console.WriteLine("Pirmoji 'ą' yra pozicijoje " +
                  raidės.IndexOf(paieška[0]));
Console.WriteLine("Pirmoji 'ą', pradedant nuo 5, yra " +
                  raidės.IndexOf('ą', 5));
Console.WriteLine("Pirmoji 'k', pradedant nuo 20 ir 7 pozicijose, yra "
                  + raidės.IndexOf('k', 20, 7));
```

Eilučių metodų pavyzdžiai 8

```
Console.WriteLine();
Console.WriteLine("Paskutinė 'm' yra pozicijoje " +
                  raidės.LastIndexOf('m'));
Console.WriteLine("Paskutinė 'm', iki 21 pozicijos, yra " +
                  raidės.LastIndexOf('m', 21));
Console.WriteLine("Paskutinė 'k', baigiant 15 ir 7 pozicijose, yra "
                  + raidės.LastIndexOf('k', 15, 7));
```

```
Pirmoji 'ą' yra pozicijoje 1
Pirmoji 'ą', pradedant nuo 5, yra 33
Pirmoji 'k', pradedant nuo 20 ir 7 pozicijose, yra -1

Paskutinė 'm' yra pozicijoje 31
Paskutinė 'm', iki 21 pozicijos, yra -1
Paskutinė 'k', baigiant 15 ir 7 pozicijose, yra 11
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 9

```
string raidės = "aąabcdeęė kkk fcdc bghiį 123 jklmną abccccc kloiiii";
Console.WriteLine("Pirmoji \"abc\" yra pozicijoje " +
                  raidės.IndexOf("abc"));
Console.WriteLine("Pirmoji \"abc\", pradedant nuo 7, yra " +
                  raidės.IndexOf("abc", 7));
Console.WriteLine("Pirmoji \"ab\", pradedant nuo 5 ir 20 pozicijose, yra "
                  + raidės.IndexOf("ab", 2, 10));
```

Eilučių metodų pavyzdžiai 10

```
Console.WriteLine();
Console.WriteLine("Paskutinė \"abc\" yra pozicijoje " +
                  raidės.LastIndexOf("abc"));
Console.WriteLine("Paskutinė \"abc\", iki 21 pozicijos, yra "
                  + raidės.LastIndexOf("abc", 21));
Console.WriteLine("Paskutinė \"ab\", baigiant 11 ir 5 pozicijose, yra "
                  + raidės.LastIndexOf("ab", 11, 5));
```

```
Pirmoji "abc" yra pozicijoje 2
Pirmoji "abc", pradedant nuo 7, yra 36
Pirmoji "ab", pradedant nuo 5 ir 20 pozicijose, yra 2

Paskutinė "abc" yra pozicijoje 36
Paskutinė "abc", iki 21 pozicijos, yra 2
Paskutinė "ab", baigiant 11 ir 5 pozicijose, yra -1
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 11

```
string raidės = "aąabcdeęė kkk fcdc bghiį 123 jklmną abccccc kloiiii";
char[] paieška = { 'ą', 'd', '1' };
Console.WriteLine("Pirmoji iš 'ą', 'd', '1' yra pozicijoje "
                  + raidės.IndexOfAny(paieška));
Console.WriteLine("Pirmoji iš 'ą', 'd', '1', pradedant nuo 7, yra "
                  + raidės.IndexOfAny(paieška, 7));
Console.WriteLine("Pirmoji iš 'ą', 'd', '1', pradedant nuo 12 ir 10 "
                  + "pozicijų, yra " + raidės.IndexOfAny(paieška, 12, 10));
```

Eilučių metodų pavyzdžiai 12

```
Console.WriteLine();
Console.WriteLine("Paskutinė iš 'ą', 'd', '1,' yra pozicijoje "
                  + raidės.LastIndexOfAny(paieška));
Console.WriteLine("Paskutinė iš 'ą', 'd', '1', baigiant 8, yra "
                  + raidės.LastIndexOfAny(paieška, 8));
Console.WriteLine("Paskutinė iš 'ą', 'd', '1,', baigiant 20 ir 8 "
                  + "pozicijų, yra " + raidės.LastIndexOfAny(paieška, 20, 8));
```

```
Pirmoji iš 'ą', 'd', '1' yra pozicijoje 1
Pirmoji iš 'ą', 'd', '1', pradedant nuo 7, yra 16
Pirmoji iš 'ą', 'd', '1', pradedant nuo 12 ir 10 pozicijų, yra 16

Paskutinė iš 'ą', 'd', '1,' yra pozicijoje 34
Paskutinė iš 'ą', 'd', '1', baigiant 8, yra 5
Paskutinė iš 'ą', 'd', '1,', baigiant 20 ir 8 pozicijų, yra 16
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 13

```
string tekstas = "aąbcdeęė fcdc bghiį jklmną";
Console.WriteLine("Eilutės dalis nuo 10 pozicijos \"" +
                  tekstas.Substring(10) + "\"");
Console.WriteLine("Eilutės dalis nuo 0 pozicijos 7 simboliai \""
                  + tekstas.Substring(0, 7) + "\"");
```

```
Eilutės dalis nuo 10 pozicijos "cdc bghiį jklmną"
Eilutės dalis nuo 0 pozicijos 7 simboliai "aąbcdeę"
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 14

```
string eil1 = "Vyksta paskaita";
string eil2 = "Metų pabaiga";
string eil3 = " ,.,Artėja sesija:;   ";
char[] simboliai = { ' ', ',', '.', ';', ':' };
Console.WriteLine("eil1 = " + "\"" + eil1 + "\"\n" +
                  "eil2 = " + "\"" + eil2 + "\"\n" +
                  "eil3 = " + "\"" + eil3 + "\"");
Console.WriteLine("\nPakeisti 'ė' su 'Ė' eil3" +
                  eil3.Replace('ė', 'Ė') + "\"");
Console.WriteLine("\neil2.ToUpper() = \"" + eil2.ToUpper() +
                  "\"\nneil3.ToLower() = \"" + eil3.ToLower() + "\"");
```

```
eil1 = "Vyksta paskaita"
eil2 = "Metų pabaiga"
eil3 = " ,.,Artėja sesija:;   "

Pakeisti 'ė' su 'Ė' eil3 ,.,ArtĖja sesija:;   '

eil2.ToUpper() = "METŲ PABAIGA"
neil3.ToLower() = " ,.,artėja sesija:;   "
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 15

```
string eil3 = " ,.,Artėja sesija:;   ";
char[] simboliai = { ' ', ',', '.', ';', ':' };
Console.WriteLine("\neil3 po Trim = \"" + eil3.Trim() + "\"");

Console.WriteLine("\neil3 po Trim = \"" + eil3.Trim(simboliai) + "\"");
```

```
eil3 po Trim = ",.,Artėja sesija:;"

eil3 po Trim = "Artėja sesija"
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 16

```
string eil1 = "Vyksta paskaita.";
string[] skyrikliai = new string[] {".", ",", ";", " "};
string[] rezultatai;
rezultatai = eil1.Split(skyrikliai, StringSplitOptions.RemoveEmptyEntries);
Console.WriteLine("Elementų kiekis rezultatų masyve: {0}", rezultatai.Length);

foreach (string s in rezultatai)
    Console.WriteLine( s );
Console.WriteLine();
char [] char_mas = { '.', ',', ' ', ';', ':' };
rezultatai = eil1.Split(char_mas, StringSplitOptions.RemoveEmptyEntries);
foreach (string s in rezultatai)
    Console.WriteLine(s);
```

```
Elementų kiekis rezultatų masyve: 2
Vyksta
paskaita

Vyksta
paskaita
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 17

```
string eil1 = "Vyksta paskaita.";
string eil2, eil3;
eil2 = eil1.Remove(7);
Console.WriteLine(eil2);
eil2 = eil1.Remove(0, 7);
Console.WriteLine(eil2);
eil3 = eil1.Insert(7, "objektinio programavimo ");
Console.WriteLine(eil3);
```

```
Vyksta
paskaita.
Vyksta objektinio programavimo paskaita.
Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Klasė `StringBuilder` *ir jos metodai*

Konstruktoriai

Keičiamos (mutable) eilutės. 6 konstruktoriai. Naujas faktorius – talpa (capacity). Dažniausiai naudojami šie 3:

```
StringBuilder bufer1 = new StringBuilder();
StringBuilder bufer2 = new StringBuilder(7);
StringBuilder bufer3 = new StringBuilder("Vyksta paskaita");
Console.WriteLine("bufer1 = " + "\"" + bufer1 + "\"\n" +
                  "bufer2 = " + "\"" + bufer2 + "\"\n" +
                  "bufer3 = " + "\"" + bufer3 + "\"\n");
```

```
bufer1 = ""
bufer2 = ""
bufer3 = "Vyksta paskaita"
```

Eilučių metodai 1

Eil.Length – eilutės ilgis;

Eil.Capacity – eilutės talpa;

Eil.EnsureCapacity(kiek) – talpos parametro nustatymas;

Eil.Append(kint) – eilutės papildymas. kint - bool, char, int, double, string tipo kintamasis arba char tipo masyvas;

Eil.AppendFormat(string_eil, mas) – mas **Object** tipo masyvas; bazinio arba eilutės tipo kintamasis;

Eil.Insert(vieta, kint) – įterpimas į eilutę. kint - bool, char, int, double, string tipo kintamasis arba char tipo masyvas;

Eil.Remove(pr, kiek) – šalina iš eilutės simbolius.
pr – pradinis adresas; kiek – kiekis;

Eilučių metodai 2

Eil.Replace(simb1, simb2) – visus simb1 simbolius keičia simboliais simb2;

Eil.Replace(gr1, gr2) – visas gr1 simbolių grupes keičia simbolių grupėmis gr2;

Eil.Replace(simb1, simb2, pr, kiek) – visus simb1 simbolius keičia simboliais simb2; pr – pradžios indeksas, kiek - ilgis;

Eil.Replace(gr1, gr2, pr, kiek) – visas gr1 simbolių grupes keičia simbolių grupėmis gr2; pr – pradžios indeksas, kiek – ilgis.

Eilučių metodų pavyzdžiai 1

```
StringBuilder eil3 = new StringBuilder("Vyksta paskaita");
Console.WriteLine("eil3 = " + "\"" + eil3 + "\"\n" +
                  "Ilgis = " + eil3.Length +
                  "\nTalpa = " + eil3.Capacity);
eil3.EnsureCapacity(50);
Console.WriteLine("\nNauja talpa = " + eil3.Capacity);
eil3.Length = 7;
Console.WriteLine("Naujas ilgis = " + eil3.Length);
for (int i = 0; i < eil3.Length; ++i)
    Console.Write(eil3[i]);
Console.WriteLine();
```

```
eil3 = "Vyksta paskaita"
Ilgis = 15
Talpa = 16

Nauja talpa = 50
Naujas ilgis = 7
Vyksta
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 2

```
string strReiksme = "Paskaita";
char[] chrMasyvas = { 'R', 'y', 't', 'a', 's' };
bool booReiksme = true;
char chrReiksme = '!';
int intReiksme = 101;
double douReiksme = 3.14;
StringBuilder eil4 = new StringBuilder();
eil4.Append(strReiksme); eil4.Append(" ");
eil4.Append(chrMasyvas); eil4.Append(" ");
eil4.Append(booReiksme); eil4.Append(" ");
eil4.Append(chrReiksme); eil4.Append(" ");
eil4.Append(intReiksme); eil4.Append(" ");
eil4.Append(douReiksme);
Console.WriteLine("eil4 = " + "\"" + eil4.ToString() + "\"\n");
```

```
eil4 = "Paskaita Rytas True ! 101 3,14"

Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 3

```
StringBuilder eil5 = new StringBuilder();
string eil1 = "Šie {0} yra {1}.\n";
object[] objMas = new object[2];
objMas[0] = "metai";
objMas[1] = 2015;
eil5.AppendFormat(eil1, objMas);
string eil2 = "Skaičius {0:d3}.\n" +
              "Išlygintas į dešinę su {0, 5} tarpais.\n" +
              "Išlygintas į kairę su {0, -5} tarpais.";
eil5.AppendFormat(eil2, 6);
Console.WriteLine(eil5.ToString());
```

```
Šie metai yra 2015.
Skaičius 006.
Išlygintas į dešinę su     6 tarpais.
Išlygintas į kairę su 6     tarpais.
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 4

```
string strReiksme = "Paskaita";
char[] chrMasyvas = { 'R', 'y', 't', 'a', 's' };
bool booReiksme = true;
char chrReiksme = '!';
int intReiksme = 101;
double douReiksme = 3.14;
StringBuilder eil6 = new StringBuilder();
eil6.Insert(0, strReiksme); eil6.Insert(0, " ");
eil6.Insert(0, chrMasyvas); eil6.Insert(0, " ");
eil6.Insert(0, booReiksme); eil6.Insert(0, " ");
eil6.Insert(0, chrReiksme); eil6.Insert(0, " ");
eil6.Insert(0, intReiksme); eil6.Insert(0, " ");
eil6.Insert(0, douReiksme); eil6.Insert(0, " ");
Console.WriteLine("Eilutė po įterpimų \n" + eil6);
eil6.Remove(11, 1);
eil6.Remove(4, 4);
Console.WriteLine("Eilutė po pašalinimo \n" + eil6);
```

```
Eilutė po įterpimų
 3,14 101 ! True Rytas Paskaita
Eilutė po pašalinimo
 3,11 !True Rytas Paskaita
Press any key to continue . . .
```

Eilučių metodų pavyzdžiai 5

```
StringBuilder eil7 = new StringBuilder("Geras rytas");
StringBuilder eil8 = new StringBuilder("Good morning");
Console.WriteLine("Eilutės prieš pakeitimus\n" + eil7 + "\n" + eil8);
eil7.Replace("Geras", "Labas");
eil8.Replace('o', 'O', 0, 5);
Console.WriteLine("\nEilutės po pakeitimų\n" + eil7 + "\n" + eil8);
```

```
Eilutės prieš pakeitimus
Geras rytas
Good morning

Eilutės po pakeitimų
Labas rytas
GOOd morning
Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Teksto analizė ir redagavimas

Teksto analizės ir redagavimo uždaviniai

- Teksto **analizė** – tai teksto charakteristikų radimas, nekeičiant paties teksto.
- Teksto **redagavimas** – tai kokių nors teksto dalių perkėlimas į kitą vietą, šalinimas, pakeitimas, sukeitimas, įterpimas ir panašūs veiksmai.

ktu informatikos fakultetas 1922

Tekstas, eilutės, simboliai

- **Tekstas** saugomas tekstiniame faile.
- Tekstas sudarytas iš eilučių.
- **Eilutė** sudaryta iš simbolių.
- Eilutės simboliai gali būti grupuojami į **grupes** ir **žodžius**.

Simbolių grupės, žodžiai, skyrikliai

- **Simbolių grupė** – tai iš eilės užrašytų simbolių seka, turinti kokį nors išskirtinį požymį (nurodytas pirmas, paskutinis simbolis ir pan.).
- **Žodis** – tai simbolių grupė, apribota iš abiejų pusių vienu arba keliais skyrikliais (skyriklių gali nebūti prieš 1-ą žodį ir už paskutinio žodžio).
- **Skyrikliai** – tai simboliai, nenaudojami žodžiams sudaryti, bet naudojami žodžiams atskirti.

ktu informatikos fakultetas 1922

Teksto eilutės

- **Teksto eilutė** – tai teksto simbolių seka nuo vieno simbolio **'\n'** iki kito simbolio **'\n'** (gali nebūti teksto pradžioje ir gale).
- Tekstas eilutėse **skirstomas į žodžius**, kuriuos skiria skyrikliai.
- Laikysime, kad žodis **prasideda ir baigiasi toje pačioje** teksto eilutėje.

ktu informatikos fakultetas 1922

Žodžiai ir skyrikliai

- Nereikia žodžio suprasti literatūriškai, kad jį sudaro tik raidės.
- Žodis gali būti sudarytas vien tik iš raidžių, vien tik iš skaitmenų arba iš raidžių ir skaitmenų. Skaitmuo – tai ne skaičius.
- Skyrikliai: tarpas (' '), kablelis (','), taškas ('.'), naujos eilutės simbolis ('\n') ir kiti spec. ženklai.

Teksto analizės problemos

- Žodžių išdėstymas teksto eilutėse nereguliarus.
- Eilučių bei žodžių ilgiai nevienodi.
- Iš eilės gali būti ir keli skyrikliai.

ktu informatikos fakultetas 1922

Tekstinio failo apribojimai

- **Skaitoma** ta teksto eilutė, kuri yra tolimesnė už paskutinės jau perskaitytos failo eilutės.
- **Negalima skaityti** iš failo, atidaryto rašymui.

- Nauja teksto eilutė visada **rašoma** į failo pabaigą.
- **Negalima rašyti** į failą, atidarytą skaitymui.

Teksto analizės atlikimo būdai

- Tekstas analizuojamas nuosekliai, eilutė po eilutės.
- Eilutėse tekstas analizuojamas po simbolį, išskiriant žodžius ir skyriklius.
- Eilutės simbolius galima nagrinėti po vieną, iš kurių sudaromi žodžiai. Tai neefektyvus, daug pastangų reikalaujantis ir sudėtingas būdas.
- Žymiai geriau naudoti **string** klasės metodus, kurie skirti darbui su simbolių eilute.

Teksto analizės lygmenys

3 lygmenys:

– failo;

– eilutės;

– žodžio.

Programa tampa žymiai lengviau skaitoma ir suprantama.

Teksto analizės lygmenų detalės

- **Failo lygmenyje**: atidaryti failą, sudaryti ciklą, kurio viduje skaitomos eilutės, atlikti operacijas su eilutėmis, uždaryti failą.
- **Eilutės lygmenyje**: sudaryti ciklą, kurio viduje išskiriami žodžiai ir atliekami veiksmai su žodžiais.
- **Žodžio lygmenyje**: atlikti veiksmus su atskirais žodžio simboliais.

Teksto apdorojimo eilutėmis algoritmas

Teksto apdorojimo prielaidos

- Teksto apdorojimo algoritmas turi tikti bet kokio ilgio tekstui analizuoti ir redaguoti.
- Laikoma, kad tekstas gali būti tokio ilgio, kad viso jo surašymas į operatyviąją atmintį (pvz., simbolių ar eilučių masyvą) yra sunkiai įmanomas.
- Dažniausiai pasirenkamas variantas – operatyviojoje atmintyje laikyti vieną teksto eilutę.

Apdorojimo eilutėmis algoritmas

Pradžia ApdorotiTekstą

    Kol ne teksto pabaiga

        Skaityti eilutę

        Analizuoti ir redaguoti (jei reikia) eilutę

        Rašyti eilutę (jei reikia)

Pabaiga

Leksinė, sintaksinė, semantinė analizė

ktu informatikos fakultetas 1922

Sąvokos

- **Leksinė analizė:** tekstas yra suskaidomas į smulkiausius kalbos elementus (pvz., programavimo kalbos identifikatorius, operatorius ir kt.).
- **Sintaksinė analizė:** nagrinėjama, ar pateiktas tekstas atitinka kalbos sintaksės taisykles (pvz., programavimo kalboje ar tinkamas skliaustų skaičius, ar rezervuoti žodžiai nenaudojami kaip identifikatoriai ir pan.).
- **Semantinė analizė:** nagrinėjama, ar pateiktas tekstas atitinka kalbos semantikos taisykles (pvz., programavimo kalboje ar klasė turi tokį elementą, į kokį kreipiamasi, ar priskiriama tinkamo tipo reikšmė ir pan.).

ktu informatikos fakultetas 1922

Teksto analizės ciklas

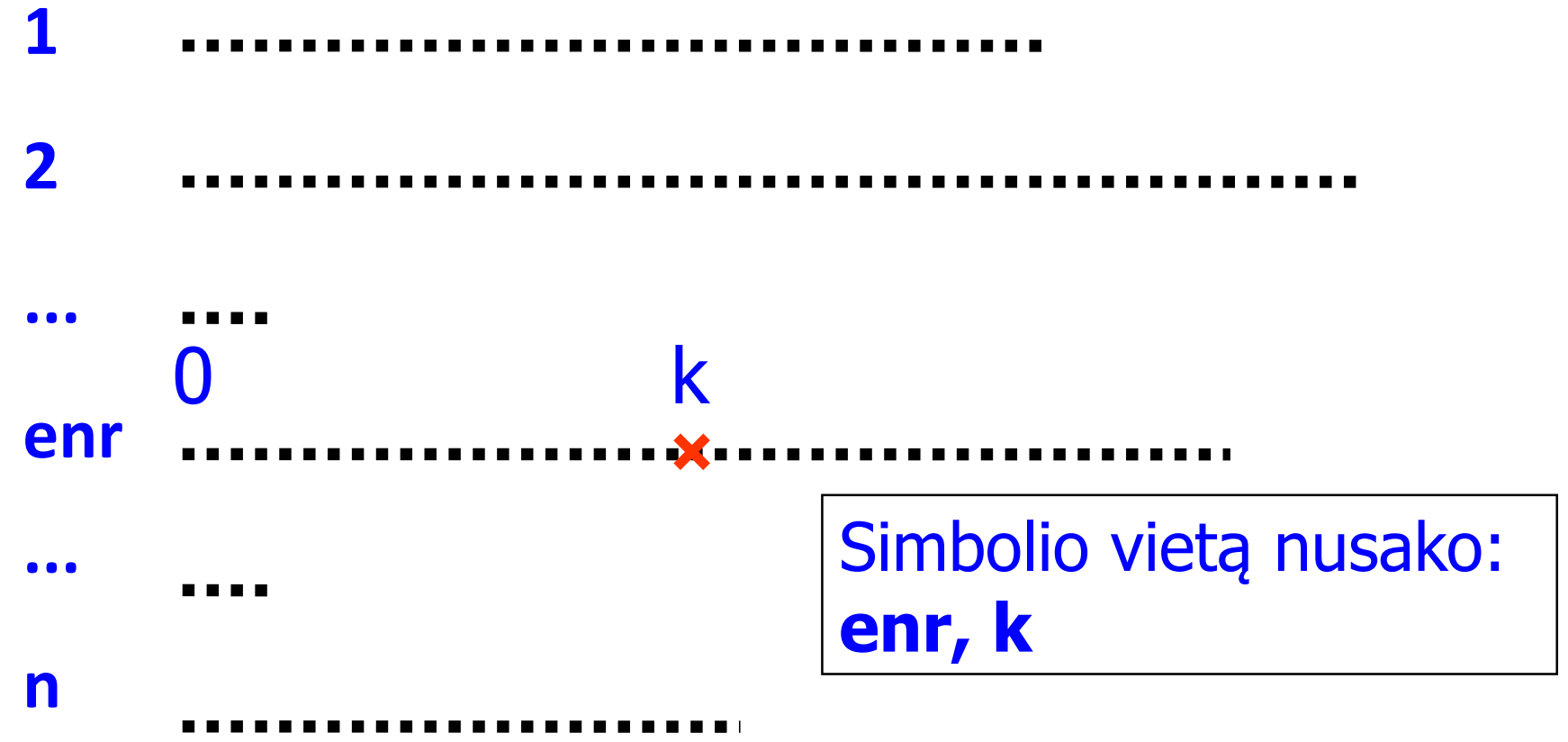
```
static void SkaitoRaso(string fs, string fr)      {
 using (var frr = File.CreateText(fr))
 {
  using (StreamReader reader = new StreamReader(fs,
         Encoding.GetEncoding(1257)))
  {
      string line;
      while ((line = reader.ReadLine()) != null)
      {
          //  teksto operacijos
          frr.WriteLine(line);
      }
   }
 }
}
```

ktu informatikos fakultetas 1922

Teksto analizės veiksmai

- **nustatyti** tam tikrus **skaitinius** teksto **parametrus** (eilučių, žodžių skaičius ir pan.);
- **atpažinti simbolius** ir rasti jų vietą tekste;
- **atpažinti** tam tikras **simbolių sekas** (žodžius, sakinius ir pan.) ir rasti jų vietą tekste;
- **palyginti simbolių sekas** pagal tam tikrus požymius (kodus, ilgį ir t. t.).

Simbolio vieta faile



Simbolio vietą nusako:
enr, k

ktu informatikos fakultetas 1922

Simbolių atpažinimo pavyzdys 1

```
// Faile fv randa 1-o skaitmens eilutės nr. enr ir indeksą k eilutėje
static void RastPirmąSk(string fv, out int enr, out int nr)
{
    using (StreamReader reader = new StreamReader(fv,
           Encoding.GetEncoding(1257)))
    {
        string line;
        char[] sk = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };
        enr = -1; int n = 0; nr = -1;
        while (((line = reader.ReadLine()) != null) && enr == -1)
        {
            n++;
            nr = line.IndexOfAny(sk);
            if (nr > -1) enr = n;
        }
    }
}
```

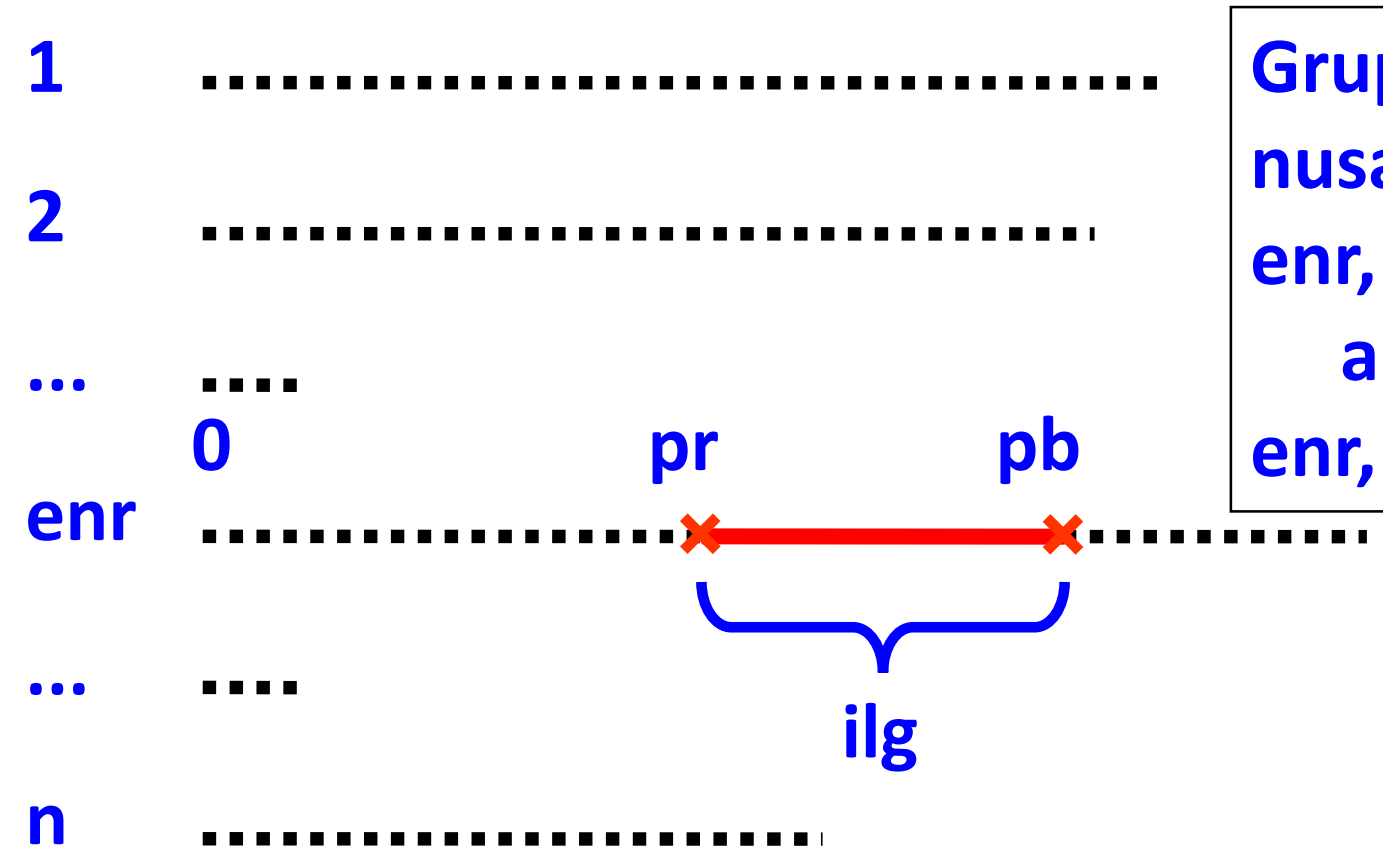
Simbolių atpažinimo pavyzdys 2

```
int enr, nr;
// Grąžina pirmojo skaitmens indeksą eilutėje eil,
// jei skaitmenų nėra, - grąžina (-1)
RastPirmąSk(CFd1, out enr, out nr);
Console.WriteLine("enr = {0} nr = {1}", enr, nr);
```

	0123456789012345678901234567890123456789012345678
1	Minutės pasisekimas apmoka metų nesėkmę. (Robert Browning)
2	Kol nepradėsime, tol nesužinosime. (Howard Zinn)
3	Kur maža žodžių, ten jie turi galią. (Viljamas Šekspyras)
4	Dar niekas nepasiklydo tiesiame kelyje. (Indų patarlė)
5	Kantrybė yra pagrindinis raktas į sėkmę. (Bilas Geitsas)
6	Sunkumuose visada slypi galimybės. (Albertas Einšteinas)
7	pi = 3,14159 (matematika)
8	Nepasikliauk pinigais, pasikliauk protu. (Kinų išmintis)

```
enr = 7 nr = 5
Press any key to continue . . .
```

Simbolių grupės vieta faile



Grupės vietą nusako:
enr, pr, pb
arba
enr, pr, ilg

Simbolių grupių atpažinimas

```
// Faile fv randa duotos grupės gr eilutės numerį enr ir pradžią pr eilutėje.
// Jei grupių nėra, - grąžina (-1)
static void RastiGrupęFaile(string fv, string gr, out int enr, out int nr)
{
    using (StreamReader reader = new StreamReader(fv,
           Encoding.GetEncoding(1257)))
    {
        string line;
        enr = -1; int n = 0; nr = -1;
        while (((line = reader.ReadLine()) != null) && enr == -1)
        {
            n++;
            nr = line.IndexOf(gr);
            if (nr > -1) enr = n;
               }
           }
      }
```

```
grupė = ne enr = 1 nr = 32
Press any key to continue . . .
```

Žodžio nusakymas eilutėje ir tekste

- **Žodį** (kaip ir simbolių grupę) nusako eilutės numeris, pradžios simbolio numeris eilutėje ir simbolių skaičius žodyje (**enr**, **pr**, **ilg**).
- Žodžiai (pirmas ir paskutinis – nebūtinai) iš abiejų pusių **ribojami skyrikliais**.

Žodžių išskyrimas eilutėje

```
char[] skyrikliai = { ' ', '.', ',', '!', '?', ':', ';', '(', ')', '\t' };
```

```
static void Zodziai(string eilute, char[] skyrikliai)
{
    string[] parts = eilute.Split(skyrikliai,
                     StringSplitOptions.RemoveEmptyEntries);
    foreach (string zodis in parts)
        Console.Write(" {0}", zodis);
    Console.WriteLine();
}
```

Žodžių išskyrimas faile 1

```
// Visas failas nuskaitomas į masyvą.
static void Apdoroti(string fv, char[] skyrikliai)
{
    string[] lines = File.ReadAllLines(fv, Encoding.GetEncoding(1257));
    foreach (string line in lines)
        if (line.Length > 0)
        {
            Zodziai(line, skyrikliai);
        }
}
```

```
char[] skyrikliai = { ' ', '.', ',', '!', '?', ':', ';', '(', ')', '\t' };
Apdoroti(CFd1, skyrikliai);
```

Žodžių išskyrimas faile 2

```
// Skaitoma iš failo po vieną eilutę.
static void Apdoroti1(string fv, char[] skyrikliai)
{
    using (StreamReader reader = new StreamReader(fv,
           Encoding.GetEncoding(1257)))
    {
        string line;
        while (((line = reader.ReadLine()) != null))
        {
            if (line.Length > 0)
                Zodziai(line, skyrikliai);
        }
    }
}
```

```
char[] skyrikliai = { ' ', '.', ',', '!', '?', ':', ';', '(', ')', '\t' };
Apdoroti1(CFd1, skyrikliai);
```

Atsakymų pavyzdys

```
Minutės pasisekimas apmoka metų nesėkmę Robert Browning
Kol nepradėsime tol nesužinosime Howard Zinn
Kur maža žodžių ten jie turi galią Viljamas Šekspyras
Dar niekas nepasiklydo tiesiame kelyje Indų patarlė
Kantrybė yra pagrindinis raktas į sėkmę Bilas Geitsas
Sunkumuose visada slypi galimybės Albertas Einšteinas
pi = 3 14159 matematika
Nepasikliauk pinigais pasikliauk protu Kinų išmintis

Press any key to continue . . .
```

Ilgiausio žodžio vietos radimas eilutėje

```
static void RastiIlgŽodįEil(string eil, char[] skyr, out int pr, out int ilg)
{
    string zodiIlg = "";
    string[] parts = eil.Split(skyr, StringSplitOptions.RemoveEmptyEntries);
    foreach (string zodis in parts)
        if (zodis.Length > zodiIlg.Length)
            zodiIlg = zodis;
        pr = eil.IndexOf(zodiIlg);
        ilg = zodiIlg.Length;
}
```

Ilgiausio žodžio vietos radimas faile 1

```
// Visas failas nuskaitomas į masyvą.
static void RastiIlgŽodįFaile(string fv, char[] skyr,
                out int enr, out int pr, out int ilg)
 {
     string[] lines = File.ReadAllLines(fv, Encoding.GetEncoding(1257));
     enr = pr = -1; ilg = 0;
     int n = 0; int pre; int ilge;
     foreach (string line in lines){
         n++;
         if (line.Length > 0){
             RastiIlgŽodįEil(line, skyr, out pre, out ilge);
             if (ilge > ilg)
             {
                 enr = n; pr = pre; ilg = ilge;
             }
         }
     } }
```

```
eilutė = 2 pradžios nr = 21  ildis = 12
Press any key to continue . . .
```

Ilgiausio žodžio vietos radimas faile 2

```
// Skaitoma iš failo po eilutę.
static void RastiIlgŽodįFaile1(string fv, char[] skyr,
                out int enr, out int pr, out int ilg)
{
    using (StreamReader reader = new StreamReader(fv,
           Encoding.GetEncoding(1257)))
    {
        enr = pr = -1; ilg = 0; int n = 0; int pre; int ilge; string line;
        while (((line = reader.ReadLine()) != null))
        {
          n++;
          if (line.Length > 0)
          {
              RastiIlgŽodįEil(line, skyr, out pre, out ilge);
              if (ilge > ilg)
                  {
                      enr = n; pr = pre; ilg = ilge;
} } } } }
```

Pavyzdys

Raskite skirtingų skaitmenų kiekį eilutėje.

```
// Randa skirtingų skaitmenų kiekį eilutėje
static int SkirtSkaitmenuSkaicius(string e)
{
    char[] balses = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
    int kiek = 0;
    for (int i = 0; i < balses.Length; i++)
        if (e.IndexOf(balses[i]) >= 0) kiek++;
    return kiek;
}
```

```
string e = "11abc222d2455555r";
int sk = SkirtSkaitmenuSkaicius(e);
Console.WriteLine("sk = {0}", sk);
```

```
sk = 4
Press any key to continue . . .
```

Eilutės redagavimas 1

Duotą eilutės žodį perkelkite į eilutės pradžią.

```
// Perkelia žodį zod į eilutės eil pradžią. pr – žodžio pradžios indeksas
static void Perkelti(ref string eil, string zod, int pr)
{
    string eil1 = eil.Remove(pr, zod.Length);
    eil = eil1.Insert(0, zod);
}
```

```
string eil = "Jonas Jonaitis dirba.";
Console.WriteLine(eil);
Perkelti(ref eil, "Jonaitis ", 6);
Console.WriteLine(eil);
```

```
Jonas Jonaitis dirba.
Jonaitis Jonas dirba.
Press any key to continue . . .
```

Eilutės redagavimas 2

Sukeiskite vietomis nurodytus žodžius.

```
// Sukeičia du žodžius: zod1 ir zod2, vietomis. pr1 - pirmo žodžio pradžia;
// pr2 - antro pradžia
static void Perkelti1(ref string eil, string zod1, string zod2, int pr1, int pr2)
{
    if (pr1 < pr2){
        string eil2 = eil.Remove(pr1, zod1.Length);
        eil = eil2.Insert(pr1, zod2);
        if (zod1.Length < zod2.Length) pr2 = pr2 + (zod2.Length - zod1.Length);
        else pr2 = pr2 - (zod1.Length - zod2.Length);
        eil2 = eil.Remove(pr2, zod2.Length); eil = eil2.Insert(pr2, zod1);
    }
    else {
        string eil2 = eil.Remove(pr2, zod2.Length);
        eil = eil2.Insert(pr2, zod1);
        if (zod1.Length < zod2.Length) pr1 = pr1 - (zod2.Length - zod1.Length);
        else pr1 = pr1 + (zod1.Length - zod2.Length);
        eil2 = eil.Remove(pr1, zod1.Length); eil = eil2.Insert(pr1, zod2);
        } }
```

Eilutės redagavimas 3

```
// Sukeičia du žodžius vietomis eilutėje
string eil = "Jonas skaito. Petras skaičiuoja. Rimas rašo.";
Console.WriteLine(eil);
Perkelti1(ref  eil, "skaito", "skaičiuoja", 6, 21);
Console.WriteLine(eil);
```

```
Jonas skaito. Petras skaičiuoja. Rimas rašo.
Jonas skaičiuoja. Petras skaito. Rimas rašo.
Press any key to continue . . .
```

Failo redagavimas: nurodytos failo eilutės pašalinimas, naujos įterpimas, eilučių sukeitimas vietomis faile

Tekstinio failo redagavimo specifika

Pradinis ir redaguotas tekstai turi būti saugomi skirtinguose failuose, kadangi:

–pradinio teksto failas atidaromas skaitymui (į jį **negalima rašyti**);

–kompiuterio operatyviojoje atmintyje kiekvienu momentu saugoma tik **viena** failo eilutė, t. y. prieš perskaitant naują pradinio teksto eilutę, suredaguota eilutė turi būti užrašyta į failą.

Nurodytos failo eilutės pašalinimas 1

Pradžia ŠalintiEilutę
Atidaryti fd skaitymui, fr rašymui
KopijuotiSrautoEilutes (fd, fr, enr-1)
Skaityti iš fd: eil
KopijuotilkiPabaigos (fd, fr)
Uždaryti fd, fr
Pabaiga

Pastaba. Prieš taikant šį metodą turi būti rastas šalinamos eilutės numeris **enr**.

Nurodytos failo eilutės pašalinimas 1

```
static void SkaitoRaso(string fs, string fr, int n)
{
    using (var frr = File.CreateText(fr))
    {
        using (StreamReader reader = new StreamReader(fs,
               Encoding.GetEncoding(1257)))
        {
            string line;
            for (int i = 0; i < n && ((line = reader.ReadLine()) != null); i++)
                frr.WriteLine(line);
            reader.ReadLine();
            while ((line = reader.ReadLine()) != null)
                frr.WriteLine(line);
        }
    }
}
```

Naujos eilutės įterpimas į failą 1

Pradžia ĮterptiEilutę
Kopijuoti iš **fd** į **fr** (**enr**) eilučių.
Rašyti į **fr**: **eil**.
Kopijuoti iš **fd** į **fr** likusias eilutes.
Pabaiga

Pastaba. Prieš taikant šį metodą turi būti rastas eilutės, už kurios terpiama, numeris **enr**. Taip pat turi būti žinoma eilutė **eil**, kurią reikia įterpti.

Naujos eilutės įterpimas į failą 2

```
static void SkaitoRaso1(string fs, string fr, string eil, int n)
{
    using (var frr = File.CreateText(fr))
    {
        using (StreamReader reader = new StreamReader(fs,
               Encoding.GetEncoding(1257)))
        {
            string line;
            for (int i = 0; i <= n && ((line = reader.ReadLine()) != null); i++)
                frr.WriteLine(line);
            frr.WriteLine(eil);
            while ((line = reader.ReadLine()) != null)
                frr.WriteLine(line);
        }
    }
}
```

Nurodytų eilučių sukeitimas faile 1

Pradžia SukeistiEilutes
Skaityti **fd** eilutę, kurios numeris **enr1**, į **eil1**.
Skaityti **fd** eilutę, kurios numeris **enr2**, į **eil2**.
Grįžti į **fd** pradžią.
Kopijuoti iš **fd** į **fr** (**enr1-1**) eilučių.
Rašyti į **fr: eil2**.
Kopijuoti iš **fd** į **fr** (**enr2-enr1-1**) eilučių.
Rašyti į **fr: eil1**.
Kopijuoti iš **fd** į **fr** likusias eilutes.
Pabaiga

Pastaba. Prieš taikant šį metodą turi būti rasti sukeičiamų eilučių numeriai **enr1**, **enr2** (enr1<enr2).

Nurodytų eilučių sukeitimas faile 2

```
static void SkaitoRaso2(string fs, string fr, int n1, int n2)
{
    // n1 < n2
    string line1, line2;
    using (StreamReader reader = new StreamReader(fs,
           Encoding.GetEncoding(1257)))
    {
        string line;
        for (int i = 0; i < n1 && ((line = reader.ReadLine()) != null); i++);
            line1 = reader.ReadLine();
        for (int i = n1 + 1; i < n2 && ((line = reader.ReadLine()) != null);
              i++);
            line2 = reader.ReadLine();
    }
// tęsinys kitoje skaidrėje
```

Nurodytų eilučių sukeitimas faile 3

```
using (var frr = File.CreateText(fr))
{
    using (StreamReader reader = new StreamReader(fs,
           Encoding.GetEncoding(1257)))
    {
        string line;
        for (int i = 0; i < n1 && ((line = reader.ReadLine()) != null); i++)
            frr.WriteLine(line);
        line = reader.ReadLine();
        frr.WriteLine(line2);
        for (int i = n1 + 1; i < n2 && ((line = reader.ReadLine()) != null);
              i++)
            frr.WriteLine(line);
        line = reader.ReadLine();
        frr.WriteLine(line1);
        while ((line = reader.ReadLine()) != null)
            frr.WriteLine(line);
        }  }  }
```

Teksto analizės ir redagavimo uždavinys

ktu informatikos fakultetas 1922

Užduotis

Pradinis tekstas, suskirstytas eilutėmis, užrašytas faile "Tekstas.txt". Žodžių skyrikliais yra: . , (taškas, tarpas, kablelis).

Ilgiausią teksto žodį ir jo eilutės nr., pradžią, ilgį atspausdinti į failą "Analize.txt".

Į failą "RedTekstas.txt" perrašyti pradinį tekstą be ilgiausio žodžio eilutės.

Uždavinio sprendimo algoritmas

Pradžia
dfv = "Tekstas.txt"; rfv = "RedTekstas.txt"; afv = "Analize.txt"; skyr=". ,";
RastiIlgŽodįFaile (dfv, skyr, enr, pr, ilg)
Atidaryti: fd(dfv), fr(rfv), fa(afv)
KopijuotiSrautoEilutes (fd, fr, enr-1)
Skaityti iš fd: eil; Išvesti į fa: eil.`Substring`(pr, ilg), enr, pr, ilg;
KopijuotiIkiPabaigos (fd, fr)
Pabaiga

ktu informatikos fakultetas 1922

Santrauka

- Pradinis tekstas viename faile, redaguotas – kitame.
- Į atmintį skaitoma po vieną teksto eilutę.
- Perskaityta teksto eilutė saugoma ir analizuojama **string** klasės objekte.
- Teksto analizė – ciklas per failo eilutes, ciklas eilutėje per žodžius, žodžio analizė.
- Failo eilutės pašalinimas, naujos įterpimas, eilučių sukeitimas reikalauja dviejų pradinio teksto peržiūros ciklų.

Klausimai?