8-Дәріс. Символдар мен тіркестерді өңдеу

Дәрістің мақсаты:

- C# тіліндегі string типімен жұмыс істеудің негіздерін түсіндіру.
- Жолдық айнымалыларды жариялау, біріктіру, салыстыру, ұзындығын анықтау және ішкі жолды алу тәсілдерін меңгеру.
- String және StringBuilder кластарын қолдану дағдыларын қалыптастыру.
- Жолдармен байланысты жиі қолданылатын әдістер мен қасиеттерді (Length, Substring, Replace, ToUpper, Trim, т.б.) үйрету.

Негізгі тақырыптар

1. string типі және оның ерекшеліктері.
2. Жолдарды біріктіру (конкатенация), салыстыру, индекстеу.
3. Length және Substring қасиеттері мен әдістері.
4. ToUpper(), ToLower(), Trim(), Replace(), Split() әдістерін қолдану.
5. StringBuilder класы және оның артықшылықтары (Append, Insert, Remove, Clear).
6. Жолдарды форматтау: string.Format(), $"{...}" интерполяциясы.
7. Жолдармен жұмыс істеуде өнімділікті арттыру тәсілдері.

Символдар мен тіркестерді жариялау және енгізу-шығару әдістері

Тіркестер немесе символдар тіркестері таңбаларды біріктіру арқылы құрастырылады. Мәліметтердің тіркестік (строковый) типі – string – *символдар тіркесін* анықтайды.

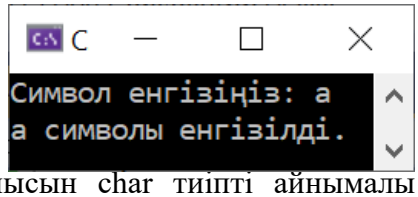
Көптеген тілдерде *тіркес* символдар жиымы ретінде қарастырылады. Ал жеке символдар стандартты char типі арқылы анықталады да, олар апострофқа ('g') алынып жазылады. Мысалы a айнымалысына g символын меншіктеу келесідей жазылады.

Char a='g';

Символдарды енгізі үшін Console класының Read әдісі қолданылады. Пернетақтадан енгізу барысында Console.Read() жеке символды int типінде қабылдайтындықтан, оны char типіне айналдыру керек болады. Мысалы келесі мысалда пернетақтадан кез келген символды енгізіп, экранға шығару көрсетілген.

```
using System;
using System.Text;
namespace Mysal
{
 class Program
  {
   static void Main(string[] args)
   { Console.OutputEncoding = Encoding.UTF8;
    char ch;
    Console.Write("Символ енгізіңіз: ");
    ch = (char)Console.Read();   // int типін char типіне келтіру
    Console.WriteLine(ch+" символы енгізілді.");
         Console.ReadKey();
   }
  }
}
```

Программаның нәтижесі:



Мұнда алдымен ch айнымалысын char типті айнымалы ретінде сипаттап, сосын Console.Read әдісі арқылы символды енгізу және char тиіне түрлендіру жүзеге асады.

C# тілінде жеке символдар басқа программалау тілдеріндегі тәрізді, 8-разрядтық кодпен (битпен) емес, уникод (Unicode) деп аталатын 16-разрядтық кодпен беріледі. Уникодтағы символдар жиыны кеңейтілген түрде бейнеленеді де, онда әлемдегі барлық табиғи тілдердің символдары толығынан қамтылған деуге болады.

Экранға шығарылатын символдардың басым бөлігін жалқы тырнақшаға алып жазу жеткілікті, бірақ кейбір символдарды мәтіндік редакторда теру, мысалы, каретканы қайтару, ерекше қиындықтар туғызады. Оған қоса, басқа бірсыпыра символдар C# тілінде арнайы қызмет атқарады (оның ішінде жалқы және қос тырнақшалар да бар), сондықтан оларды тікелей қолдана беруге болмайды.

Осындай себептерге байланысты C# тілінде арнайы символдарды басқару тізбектері, ал кейде кері сызықшадан басталатын константалар (7.1 кесте) “Ескейп-тізбектер” немесе “Басқару тізбектері” депаталатын арнайы бір топ қарастырылған. Мұндай тізбектер өздері бейнелейтін символдар орнына қолданылады.

8.1-кесте. Басқару тізбектері

Басқару тізбегі	**Сипаттамасы**
\a	Дыбыстық сигнал (қоңырау)
\b	Бір позицияға кері қайту
\f	Парақты ауыстыру (принтерде жаңа бетке көшу)
\n	Жаңа жол (келесі жолға көшу)
\r	Каретканы қайтару (принтер үшін)
\t	Көлденең (горизонталь) табуляция
\v	Тік (вертикаль) табуляция
\0	Бос символ
\'	Экранға жалқы тырнақша (апостроф) шығару
\"	Экранға қос тырнақша шығару
\\	Экранға кері қиғаш сызық шығару

Мысалы, ch айнымалысына табуляция символын меншіктеу:

```
ch = '\t';                немесе
ch = "\";       //  "\" символын меншіктеу
```

C# тілінде кең таралған бір тип – символдардың тіркестік (string) типі пайдаланылады. Осы типпен жұмыс істеудің барлық мүмкіндіктері System.String класына топтастырылған. String – бұл System.String класының бүркеншік аты. Символдық тіркестерді құрудың ең жеңіл жолы – оларды тырнақша ішіне жазып айнымалыға меншіктеу. Мысалы, s тіркесін құрайық.

```
string s ="Бұл тіркес";
```

Мұнда s айнымалысына "Бұл тіркес" атты 2 сөзден тұратын тіркес (string типіндегі символдар тіркесі) меншіктеледі.

string типіндегі объектіні char типіндегі жиым арқылы да құра аламыз:

```
char[] charray = {'т','е','s','т'};
string str = new string(charray);
```

Мұнда алдымен charray атты char тиіптегі жиым құрылып, 5 символмен инициалданды. Ары қарай str атты string тиіптегі айнымалыға charray жиымы string класының данасы ретінде меншіктеледі.

Тіркес ішіндегі жеке символды таңдап алу үшін оның индексі қолданылады. Тіркестер сиволдар жиымы болғандықтан онда тіркес символдарының индексі сандар жиымының индекстерін анықтаумен бірдей. Мысалы,

```
string  s ="Бұл тіркес";
Console.WriteLine(s[0]);
```

Программа үзіндісінде “Б” символы экранға шығады. өйткені ол бірінші орында тұр. Жиымдағы сияқты мұнда да нөмірлеу 0-ден басталады. Бірақ индекс арқылы тіркес символдарын басқаға өзгертуге болмайды. Индекс тіркестегі символдардың мәнін тексеру үшін ғана қолданылады.

Тіркес ұзындығы **Length** қасиеті арқылы табылады:

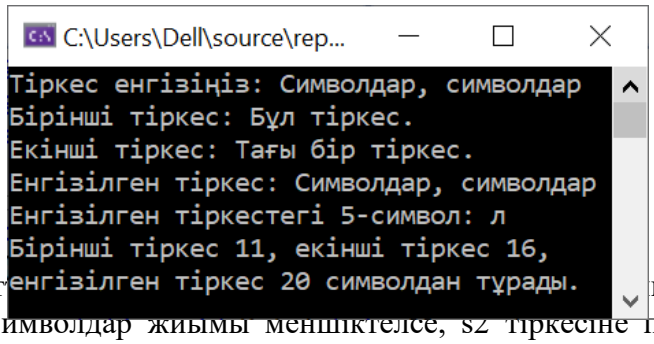
```
Console.WriteLine("L=" + s.Length);  // L=10 болады
```

Екі тіркесті немесе символдарды теңдікке тексеру үшін == операторы қолданылады. Көбінесе, егер == операторы объектіге сілтеме ретінде қолданылса, онда ол сілтемелердің бір объектіге жасалып тұрғандығын көрсетеді. Келесі мысалда тіркестерді енгізу мен шығару, индекстерін қолдануды көрсетеміз.

```
using System;
using System.Text;
namespace Mysal
{
 class Program
 {
  static void Main(string[] args)
  {
     Console.OutputEncoding = Encoding.UTF8;
     // тіркестердің мәнін беру
     char[] charray = {'Б','ұ','л',' ','т','і','р','к','е','с','.'};
     string s1 = new string(charray);
     string s2 = "Тағы бір тіркес.";
     Console.Write("Тіркес енгізіңіз: ");
     string s3 = Console.ReadLine();
     // тіркестерді шығару
     Console.WriteLine("Бірінші тіркес: " + s1);
     Console.WriteLine("Екінші тіркес: " + s2);
     Console.WriteLine("Енгізілген тіркес: " + s3);
     Console.WriteLine("Енгізілген тіркестегі 5-символ: " + s3[5]);

     Console.WriteLine("Бірінші тіркес {0}, екінші тіркес {1}," +
     "\nенгізілген тіркес {2} символдан тұрады.", s1.Length, s2.Length, s3.Length);
     Console.ReadKey();
  }
 }
}
```

Программаның нәтижесі:



Мұнда char тиіптегі charray ат малылары сипатталған. S1 тіркесіне charray символдар жиымы меншіктелсе, s2 тіркесіне программа ішінде мән беріледі. Ал s3 тіркесін Console.ReadLine әдісі арқылы енгіземіз. Console.ReadLine әдісі арқылы сандық мәліметтерді енгізуді үйренген кезде консоль енгізілген мәнді string типінде қабылдайтынын айтып, типтерді түрлендіру әдістерін қолданған болатынбыз. Сондықтан бұл әдіс енгізілген мәліметті тіркес ретінде танитындықтан тіркестерді енгізгенде ешқандай түрлендіру әдістерін қолданбаймыз. Ары қарай бұл программада s1, s2, s3 тіркестерін және олардың ұзындығын экранға шығару және s3 тіркесінің 5-индексінде орналасқан символын шығару жүзеге асады.

Ескерту: string типінде орыс алфавитіне кірмейтін қазақ ұлттық әріптері (9 әріп) дұрыс көрсетілмейді. Сондықтан қазақ әріптерін латын (немесе кириллица) әріптері арқылы бейнелеу керек.

Тіркестермен операциялар орындау

Тіркестерді өңдеу үшін String класының әдістері қолданылады. Олардың ішінде жиі қолданылатын келесі әдістерді айтуға болады:

Compare	Екі тіркесті немесе ішкі тіркестерді бір-бірімен салыстырады
Contains	Ішкі тіркестің тіркесте бар-жоғын анықтайды
Concat	Тіркестерді біріктіру үшін қолданылады
CopyTo	Белгілі бір индекстен басталатын тіркестің ішкі бөлігін көшіреді
IndexOf	Тіркестегі символдың немесе ішкі тіркестің бірінші пайда болу индексін анықтайды
Insert	Тіркеске шағын ішкі тіркесті енгізеді
Join	Тіркестер жиымының элементтерін біріктіреді
LastIndexOf	Тіркестегі символдың немесе ішкі тіркестің соңғы кездесу индексін табады
Replace	Тіркестегі символды немесе ішкі тіркесті басқа символмен немесе ішкі тіркеспен ауыстырады

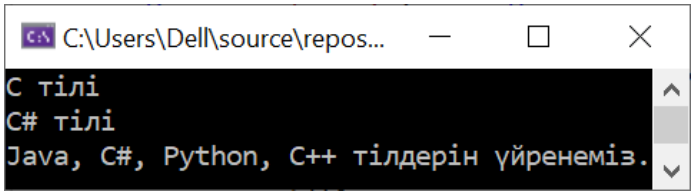
Split	Бір тіркесті бірнеше ішкі тіркестерге, тіркестер жиымына бөледі
Substring	Көрсетілген позициядан басталатын тіркестен ішкі тіркесті бөліп алады
ToLower	Тіркестегі барлық символдарды кіші әріпке түрлендіреді
ToUpper	Тіркестегі барлық символдарды бас әріпке түрлендіреді
Trim	Тіркестің алдындағы және соңындағы бос орындарды жояды.

Енді осы әдістердің қолданылуын мысалмен түсіндірейік.

Тіркестерді біріктіру + таңбасы арқылы немесе Concat, Join әдістерін қолдану арқылы жүзеге асады. Бұл әдістердің қолданылуын келесі мысалдан көруге болады:

```
using System;
using System.Text;
namespace Mysal
{ class Program
  { static void Main(string[] args)
    { Console.OutputEncoding = Encoding.UTF8;
          string s1 = "C"; string s2 = "тілі";
          string s3 = s1 + " " + s2; // тіркестерді + таңбасы арқылы біріктіру
          Console.WriteLine(s3);
          string s4 = String.Concat(s1, "# тілі"); // Concat әдісін қолданып,
          Console.WriteLine(s4);                   // тіркестерді біріктіру
          string s5 = "Java,"; string s6 = "C#,";
          string s7 = "Python,"; string s8 = "C++";
          string s9 = "тілдерін үйренеміз.";
          string[] values = new string[] { s5, s6, s7, s8, s9 };
          string s10 = String.Join(" ",values); // Join әдісін қолданып,
          Console.WriteLine(s10);               // тіркестерді біріктіру
          Console.ReadKey();
    }
  }
}
```

Программаның орындалу нәтижесі:

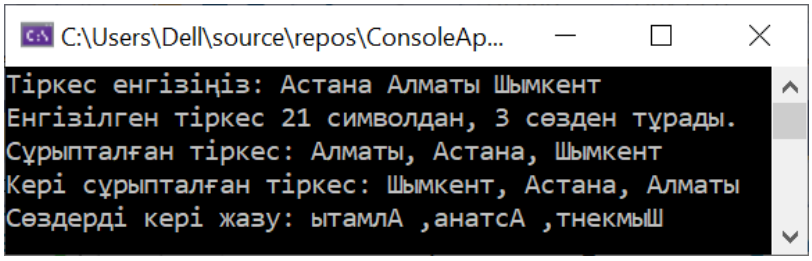


Мұнда s1 және s2 тіркестерін біріктіру үшін + таңбасы, s1 тіркесімен берілген мәтінді біріктіру үшін Concat әдісі қолданылды. Бұдан осы екі тәсілдің ешқандай айырмашылығы жоқ екенін көруге болады. Ал Join әдісін қолдану барысында алдымен тіркестерден тіркестер жиымын құрып алып, тіркестер жиымына Join әдісін қолданамыз.

Тіркестердегі символдар санын, тіркестің ұзындығын анықтау үшін Length қасиеті қолданылады. Сондай-ақ тіркестер символдар жиымы болғандықтан, тіркестерге Array класының әдістерін де қолдануға болады. Келесі мысалда тіркестерге Array класының әдістері қолданылған.

```
using System;
using System.Text;
namespace Mysal
{ class Program
  { static void Main(string[] args)
    { Console.InputEncoding = Encoding.Unicode;  // Енгізілген мәліметті тану
      Console.OutputEncoding = Encoding.Unicode; // Шығарылған мәліметті тану
      string s1, s5;
      int i, L, ss = 1; //ss айнымалысы сөздердің санын есте сақтайды
      Console.Write("Тіркес енгізіңіз: ");
      s1 = Console.ReadLine();    // s1 тіркесін енгізу
      L = s1.Length;              // енгізілген тіркестің ұзындығын табу
      for (i = 0; i < L; i++)     // тіркесті соңына дейін қарастыру
         if (s1[i] == ' ')
            ss++; // бос орын анықталса сөз санына 1 қосылады
      Console.WriteLine("Енгізілген тіркес {0} символдан, " +
                  "{1} сөзден тұрады.", L, ss);
```

```
        string[] s2 = s1.Split(new char[] {' '}); // тіркесті жиымға айналдыру
    Array.Sort(s2); // тіркестер жиымын сұрыптау
    String s3 = string.Join(", ", s2);
    Console.WriteLine("Сұрыпталған тіркес: " + s3);
    Array.Reverse(s2); // Тіркес жиымын кері бағытта жазу
    String s4 = string.Join(", ", s2);
    Console.WriteLine("Кері сұрыпталған тіркес: " + s4);
    char[] ch = s4.ToCharArray(); // тіркесті символдар жиымына айналдыру
    Array.Reverse(ch);     // жиым элементтерін кері жазу
    s5 = new string(ch);   // символдар жиымын тіркеске айналдыру
    Console.WriteLine("Сөздерді кері жазу: " + s5);
    Console.ReadKey();
  }
 }
}
```



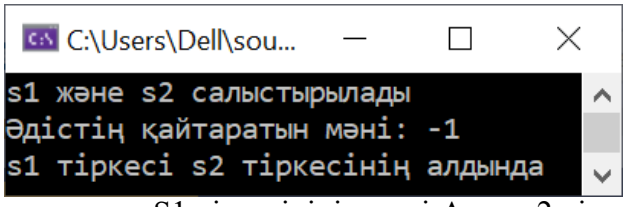
Программаның орындалу нәтижесі:

Бұл мысалда тіркестің Length қасиеті бойынша ұзындығын тауып, ұзындығы бойынша цикл құрып, жиымдағы бос орындардың санына байланысты тіркестегі сөз саны табылды. Сонымен бірге тіркесті Split() әдісі арқылы тіркестер жиымына айналдырдық. Басқа типтегі мәліметтер тәрізді тіркестер де жиымдар элементтері бола алатынын білдік. Тіркесті тіркестер жиымына Split() әдісі арқылы айналдырсақ, Тіркестер жиымын Join әдісі арқылы тіркеске айналдырамыз. Сондай-ақ тіркесті ToCharArray() әдісі арқылы символдар жиымына айналдыра аламыз. Тіркестерді жиымға айналдырған соң мысалдағы сияқты Array класының әдістерін қалауымызша қолданамыз. Мұнда Sort әдісі арқылы тіркес мүшелерін алфавит реті бойынша жаздық. Алфавит реті бойынша сұрыпталған жиымға Reverse әдісін қолдану арқылы сөздері алфавит ретіне кері жазып алдық. Сондай-ақ Reverse әдісін символдар жиымына қолданып, тіркестегі барлық символды кері бағытта жаздық

Тіркестерді салыстыру үшін String класының Compare() статикалық әдісі қолданылады. Ол тіркестердің орналасу ретін салыстырады да сандық мән қайтарады. Егер 1-тіркес алфавит бойынша 2-тіркестен жоғары тұрса, онда теріс -1 санын қайтарылады. Қарсы жағдайда, ол 1-ді қайтарады. Екі тіркес бірдей болса, онда 0 санын қайтарады. Төмендегі программа үзіндісінде екі тіркесті салыстыру көрсетілген:

```
static void Main(string[] args)
{ Console.InputEncoding = Encoding.Unicode;  // Енгізілген мәліметті тану
  Console.OutputEncoding = Encoding.Unicode; // Шығарылған мәліметті тану
  string s1 = "Asan"; string s2 = "Usen";
  int result = String.Compare(s1, s2);
  Console.WriteLine("s1 және s2 салыстырылады");
  Console.WriteLine("Әдістің қайтаратын мәні: " + result);
  if (result < 0)
    Console.WriteLine("s1 тіркесі s2 тіркесінің алдында");
  else if (result > 0)
        Console.WriteLine("s1 жолы s2 жолының артында");
     else
        Console.WriteLine("s1 және s2 тіркестері бірдей");
  Console.ReadKey();
}
```

Программаның нәтижесі:



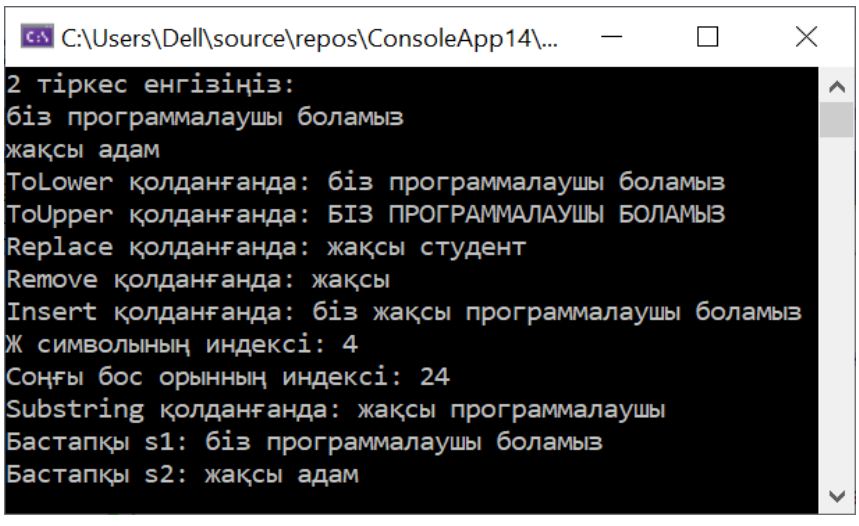
Мұнда s1 және s2 әдістерін салыстырады. S1 тіркесінінің мәні Asan, s2 тіркесінің мәні Usen сөзінен алфавит бойынша жоғары орналасқандықтан нәтижесі -1 мәнін қайтарады.

Тіркестерді өзгертуге арналған бірнеше әдіс бар. Тіркестегі әріптердің регистрін өзгерту үшін тіркестің барлық әріптерін сәйкесінше бас әріпке немесе кіші әріпке түрлендіретін

ToUpper() және ToLower() әдістерін қолдануға болады. Replace() әдісі арқылы тіркестің ішіндегі бір немесе бірнеше символды екінші бір символдармен ауыстырамыз. Бір тіркесті екінші тіркес ішіне кірістіру үшін Insert() әдісі қолданылады. Тіркес бөлігін өшіру үшін Remove() әдісі қолданылады. Тіркес бөлігін қиып алу үшін Substring() әдісі қажет. Енді осы әдістердің бәрін төмендегі мысалда көрсетейік:

```
using System;
using System.Text;
namespace Mysal
{ class Program
  { static void Main(string[] args)
    { Console.InputEncoding = Encoding.Unicode;
      Console.OutputEncoding = Encoding.Unicode;
      Char a, b;
      string s1, s2, s3, s4, s5, s6;
      int x, y;
      Console.WriteLine("2 тіркес енгізіңіз:");
      s1 = Console.ReadLine(); // біз программалаушы боламыз
      s2 = Console.ReadLine(); // жақсы адам
      // тіркестердің регистрін ауыстыру
      Console.WriteLine("ToLower қолданғанда: " + s1.ToLower());
      Console.WriteLine("ToUpper қолданғанда: " + s1.ToUpper());
      // ішкі тіркесті басқа тіркеспен ауыстыру
      s3 = s2.Replace("адам", "студент");
      Console.WriteLine("Replace қолданғанда: " + s3);
      // тіркестің бір бөлігін өшіру
      s4 = s3.Remove(6);
      Console.WriteLine("Remove қолданғанда: " + s4);
      // тіркестің арасына екінші тіркесті қосу
      s5 = s1.Insert(4, s4);
      Console.WriteLine("Insert қолданғанда: " + s5);
      // тіркестегі белгілі символдардың орнын анықтау
      a = 'ж'; b = ' ';
      x = s5.IndexOf(a);
      Console.WriteLine("Ж символының индексі: " + x);
      y = s5.LastIndexOf(b);
      Console.WriteLine("Соңғы бос орынның индексі: " + (y));
      // тіркестен ішкі тіркесті қиып алу
      s6 = s5.Substring(x, (y - x));
      Console.WriteLine("Substring қолданғанда: " + s6);
      // бастапқы тіркестерді шығару
      Console.WriteLine("Бастапқы s1: " + s1);
      Console.WriteLine("Бастапқы s2: " + s2);
      Console.ReadKey();
    }
  }
}
```

Программаның орындалу нәтижесі:



Бұл мысалда тіркестерді өңдеу үшін бірнеше әдіс қолданылған. Берілген s1 тіркеске ToUpper() және ToLower() әдістерін қолданып, регистрді өзгерттік. Replace әдісі арқылы S2 тіркесіндегі «адам» сөзін «студент» сөзіне ауыстырдық. Ары қарай Remove әдісімен «жақсы студент» болып өзгерген тіркестен, студент ішкі тіркесін өшірдік. Сосын Insert әдісін s1 тіркесіне қолданып, 4-индексінен бастап s4 (жақсы) тіркесін қосып, мәтінді «біз жақсы программалаушы боламыз» (s5) деп өзгерттік. S5 тіркесіне IndexOf және LastIndexOf әдістерін қолданып, «ж» символының алғашқы кездескендегі индексін, бос орынның соңғы кездесуіндегі индекстерін алдық. Осы индекстердің мәні бойынша тіркеске Substring әдісін қолданып, «жақсы программалаушы» деген тіркесті бөліп алдық. Бұл әдістегі 1-аргумент қай индекстен бастап бөліп алатынымыздың білдірсе, 2-аргумент бөліп алатын символдың ұзындығы.

Тіркестер тұрақтылығы. String типіндегі объектілерді өзгертуге болмайды. Бұл шектеу символдық тіркестерді тиімді түрде пайдалану мүмкіндігін береді. Сондықтан бұл мүмкіндік кей кезде артықшылыққа да айналып кетеді. Мысалы, егер бұрынғы тіркестен жаңа тіркес шығару керек болса, өзгертілген жаңа тіркес құрамыз, ал бұрынғы тіркес автоматты түрде өшіріліп кетеді де, жұмысқа ешқандай кедергі келтірмейді.

Бірақ тіркестерге сілтеме жасайтын айнымалыларды (яғни string типіндегі объектілерді) өзгертуге болады, сол себепті олар басқа объектіге сілтеме жасай алады. Дегенмен, string типіндегі объектінің мәні ол құрылғаннан кейін өзгертілмейді. Өзгертілмейтін тіркестердің кедергі болмайтынын жоғарыдағы мысалдан көруге болады. Тіркесті өзгерту кезінде басқа тіркестік айнымалыға меншіктеп отырдық. Ал s1 және s2 тіркестерін соңында шығарғанда да, өзгермей алғашқы мәндері шықты.

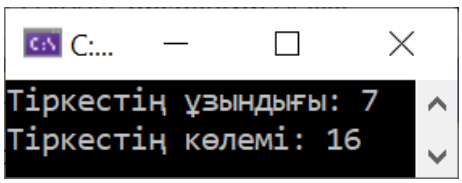
Ал тіркестерді өзгерту керек болса, StringBuilder класын қолдану керек, ол System.Text атаулар кеңістігінде анықталған. Бұл класс өзгертуге болатын тіркестік объектілер құра алады. Бірақ көбінесе C#-та программалауда StringBuilder класы емес, string типі жиі қолданылады.

StringBuilder класы

System.String класы бізге тіркестермен жұмыс істеудің ауқымды мүмкіндіктерін ұсынғанымен, оның кемшіліктері де жоқ емес. String объектісі – өзгермейтін тіркес екені алдыңғы бөлімде айтылды. String класының кез келген әдісін орындаған кезде, жүйе жадында оған жеткілікті орын бөлетін жаңа объект жасайды. Алғашқы айнымалыны жою ондай үлкен операция емес. Алайда, егер мұндай операциялар көп болса және бұл операцияларды орындау қажет мәтін де көлемді болса, онда жұмыс өнімділігіне кері әсер етеді.

Мұндай жағдайдан шығу үшін System.Text кітапханасында орналасқан .NET құрылымына жаңа StringBuilder класы қосылды. Бұл класс динамикалық тіркесті көрсетеді. Тіркесті құру кезінде StringBuilder класы компьютер жадынан осы тіркеске қажет көбірек орын бөледі:

```
StringBuilder sb = new StringBuilder("C# тілі");
Console.WriteLine($"тіркестің ұзындығы: {sb.Length}");
Console.WriteLine($"Тіркестің көлемі: {sb.Capacity}");
```



Sb айнымалысына «C# тілі» деген бастапқы тіркес меншіктелді. Бұл тіркестің ұзындығы 7 символдан тұрады. StringBuilder класында ұзындықты анықтау үшін Length қасиеті қолданылады. Сонымен қатар, тіркеске бөлінген компьютер жадының көлемі Capacity қасиеті арқылы анықталады. Мұнда тіркестің ұзындығы 7 таңба болғанымен, нақты бөлінген жады көлемі 16 символды сақтай алады.

StringBuilder класында объектіні инициалдауды әртүрлі жолмен орындауға мүмкіндік беретін басқа да тәсілдер бар. Сонымен, біз бос объект құрып, бірақ оған бөлінетін бастапқы жады көлемін орната аламыз:

StringBuilder sb = new StringBuilder(20);

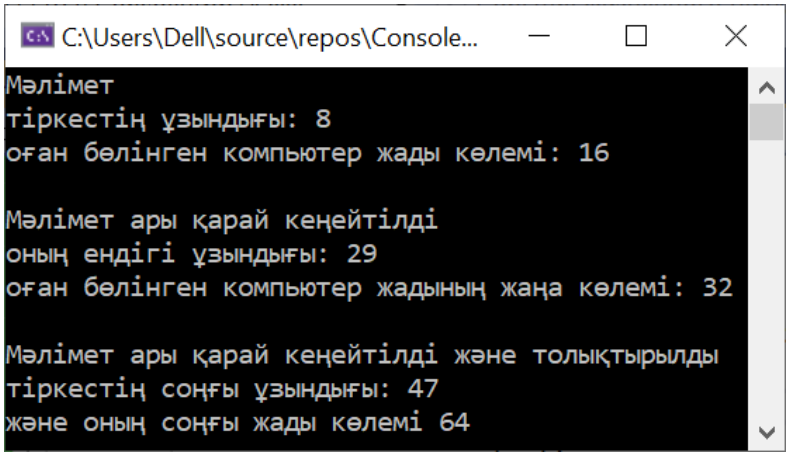
Мұнда sb объектісі құрылып, оған 20 символ сақтауға болатын жады көлемі бөлініп берілді. Егер біз объектінің максималды ұзындығын алдын ала көрсететін болсақ, онда оған тағайындалатын жады көлемін орнатып және тағы да қосымша жады бөлу арқылы өзгерістер енгізе аламыз.

Енді StringBuilder класының қолданылуын келесі мысал арқылы көрсетейік:

```
using System;
using System.Text;
namespace Mysal
{ class Program
 { static void Main(string[] args)
 { Console.OutputEncoding = Encoding.Unicode;

 StringBuilder sb = new StringBuilder("\nМәлімет");
 Console.WriteLine(sb);
 Console.WriteLine($"тіркестің ұзындығы: {sb.Length}");
 Console.WriteLine($"оған бөлінген компьютер жады көлемі: " +
 $"{sb.Capacity}");
 sb.Append(" ары қарай кеңейтілді"); // тіркес мәтінін кеңейту
 Console.WriteLine(sb);
 Console.WriteLine($"оның ендігі ұзындығы: {sb.Length}");
 Console.WriteLine($"оған бөлінген компьютер жадының жаңа көлемі: " +
 $"{sb.Capacity}");
 sb.Append(" және толықтырылды"); // тіркес мәтінін толықтыру
 Console.WriteLine(sb);
 Console.WriteLine($"тіркестің соңғы ұзындығы: {sb.Length}");
 Console.WriteLine($"және оның соңғы жады көлемі {sb.Capacity}");
 Console.ReadKey();
 }
 }
}
```

Программаның орындалу нәтижесі:



StringBuilder объектісін құрған кезде, бастапқы тіркес 16 символдан аз (\n таңбасын қосып есептегенде 8) болғандықтан, 16 таңбадан тұратын қалыпты компьютер жады бөлінеді. Содан кейін Append әдісі қолданылды – бұл әдіс тіркеске шағын ішкі тіркес (жаңа тіркес) қосады. Тіркестерді біріктіру кезінде олардың жалпы ұзындығы – 29 символ болып, жадының 8 символық бастапқы көлемінен асты. Сондықтан бастапқы жады көлемі артып, 32 таңбаға дейін жетті. Сонан кейін тіркеске тағы жаңа тіркес қосылып, тіркестің жалпы ұзындығы 32 символдан артатын болса, онда жады көлемі тағы да артатын болады. Ары қарай тіркес ұзындығы артып жатса, ұзындығына сай жады көлемі беріледі. Дәл солай Append әдісі қайтадан қолданылып, тіркестің соңғы ұзындығы 47 символ болды, бұл кезде тағы қосымша жад бөлініп, оның көлемі 64-ке жетті.

Жоғарыдағы мысалда көрсетілген Append әдісінен басқа StringBuilder класында тіркестерді өңдейтін басқа да әдістерді қолдана аламыз. Олар төмендегі кестеде келтірілген.

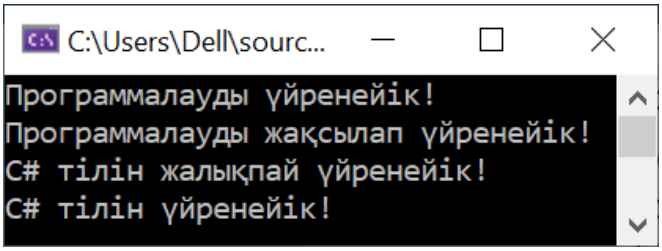
Әдіс аты	*Сипаттамасы*
Compare	екі тіркесті немесе ішкі тіркестерді бір-бірімен салыстырады
Insert	StringBuilder объектісіне, белгілі бір индекстен бастап, ішкі тіркесті енгізеді
Remove	белгілі бір индекстен басталатын берілген ұзындықтағы символдарды өшіреді
Replace	белгілі бір символды немесе ішкі тіркестің барлық қайталануын басқа символмен немесе ішкі тіркеспен ауыстырады

AppendFormat	StringBuilder объектісінің соңына ішкі тіркес қосады

Келесі мысалда осы әдістердің қолданылуы көрсетілген:

```
using System;
using System.Text;
namespace Mysal
{ class Program
  { static void Main(string[] args)
    { Console.OutputEncoding = Encoding.Unicode;
      StringBuilder sb = new StringBuilder("Программалауды үйренейік");
      sb.Append("!");
      Console.WriteLine(sb);
      // Тіркесске ішкі тіркесті кірістіру
      sb.Insert(15, "жақсылап ");
      Console.WriteLine(sb);
      // тіркестерді ауыстыру
      sb.Replace("Программалауды жақсылап", "C# тілін жалықпай");
      Console.WriteLine(sb);
      // 9-символдан бастап 9 символды өшіру
      sb.Remove(9, 9);
      // StringBuilder объектісін тіркеске айналдыру
      string s = sb.ToString();
      Console.WriteLine(s);
      Console.ReadKey();
    }
  }
}
```

Программаны орындау нәтижесі:



Мұнда берлілген sb объектісіндегі "Программалауды үйренейік" деген тіркеске Append әдісі арқылы "!" белгісі қосылды. Одан кейін тіркеске 15- индексінен бастап «жақсылап» деген тіркесті Insert әдісі арқылы қосты. Replace әдісі арқылы «Программалауды жақсылап» деген тіркесті «C# тілін жалықпай» тіркесімен ауыстырды. Ары қарай Remove әдісі арқылы тіркестің 9-индексінен бастап 9 символды өшірді. Программаның соңында sb объектісін string типіне ауыстырып экранға шығарды.

Тіркестерді оңдеу барысында String және StringBulder кластарын қолданудың өзіндік ерекшеліктері бар. Тіркестерге аздаған өзгерістер мен операциялар қолданғанда, біріктіру операцияларының бекітілген санын орындау кезінде, тіркесте IndexOf немесе StartsWith сияқты ауқымды іздеу әдістерін (StringBuilder класында бұл әдістер жоқ) қолдану қажет болған кездерде String класы пайдаланылады. Ал StringBuilder класы программаны орындау кезінде тіркестерге көптеген өзгерістер енгізу және жасайтын операциялар саны белгісіз болғанда, көптеген ұқсас операциялар жасау қажет болғанда қолданылады.

Бақылау сұрақтары

1. string типі қандай типке жатады және оның ерекшелігі неде?
2. Жолды біріктірудің (+, Concat) және форматтаудың ($, Format) айырмашылығы қандай?
3. Substring және Replace әдістері қалай жұмыс істейді?
4. StringBuilder класының басты артықшылығы неде?
5. ToUpper() және ToLower() әдістері қандай жағдайларда қолданылады?
6. Жолдағы бос орындарды алып тастау үшін қандай әдіс қолданылады?
7. Жолдың ұзындығын қалай анықтауға болады?
8. Split() әдісінің жұмыс принципін түсіндіріңіз.
9. Жолдармен жұмыс кезінде жады үнемділігін арттыру тәсілдері қандай?
10. Мәтінмен жұмыс кезінде StringBuilder қолданудың мысалын келтіріңіз.

Ұсынылатын әдебиеттер

1. Колдаев, В.Д. Основы алгоритмизации и программирования: Уч. пособие/ под ред. проф. Л.Г.Гагариной. -М.: ИД «ФОРУМ»: ИНФРА-М,2015.-416 с.
2. Бөрібаев Б. Компьютердің арифметикалық негіздері: Жоғары оқу орындары студенттеріне арналған оқу құралы. -Алматы: Қазақ университеті, 2009. -80 б.
3. Бөрібаев Б. Алгоритмдеу, мәліметтер құрылымы және программалау тілдері: оқулық. – Алматы: Қазақ университеті, 2012. -235 б.
4. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. -М.: Мир, 2011.-360с.
5. Бөрібаев Б., Абдрахманова М. C# тілінде программалау (практикалық курс): оқу құралы –Алматы: Қазақ университеті, 2018. -258 б.
6. Troy Dimes. C# Programming for Beginners. Paperback – January 25, 2015.
7. Шилдт Г. C# 4.0: полное руководство. М.: ООО ”И.Д.Вильямс”, 2017. -1056 с.
8. Джуст В. Разработка обслуживаемых программ на языке C#. СПб.: Изд. дом «Питер», 2017.
9. Шарп Д. Microsoft Visual C#. Подроб. руководство. 8-е изд.С-Петербург: Изд. дом «Питер», 2016.
10. A. Troelsen, P. Japikse. Pro C# 7: With .NET and .NET Core. – Apress, 2017. -1372 p.