

Zeichenketten in C#

Der Datentyp char

Einzelne Zeichen werden in C# mit dem Datentyp `char` (sprich: Character) gespeichert. Werte von `char`-Variablen werden innerhalb des Programmtextes in Hochkommas eingeschlossen (z.B. 'a', 'x', '?', '2'). **Achtung:** '2' stellt das Zeichen 2 dar, nicht die Zahl 2.

Der Datentyp String

Im Gegensatz zu `char` ist ein `String` eine Aneinanderreihung von einzelnen Zeichen einer beliebigen Länge. Strings deklariert man als `String x`;

Werte von `String`-Variablen werden innerhalb des Programmtextes in Anführungszeichen eingeschlossen, z.B. "Meier", "Inf-12-Gk", "12&abc\$7%?", "hallihallo".

Auf einzelne Zeichen in einem `String` greift man mit eckigen Klammern zu. In den Klammern steht die Position des gewünschten Zeichens, die von 0 gezählt wird. `s[2]` liefert für einen `String s`, der das Wort "Wurm" enthält, das Zeichen 'r'.

Zeichenketten haben eine große Anzahl von Methoden und Eigenschaften. Einige davon sind:

<code>n=s.Length;</code>	n: int s: String	Mit Hilfe der Eigenschaft <code>Length</code> wird die Länge des Strings <code>s</code> bestimmt, d.h. die Anzahl der Zeichen, aus denen <code>s</code> besteht. Beispielsweise erhält man für <code>s= "Turm"</code> ; durch <code>s.Length</code> den Wert 4. <i>Nach Length steht keine Klammer!</i>
<code>z=s.Remove(p, n);</code>	p, n: int s, z: String	Beim <code>String s</code> werden ab der Position <code>p</code> <code>n</code> Zeichen gelöscht (höchstens aber bis zum Ende des Strings) und als Ergebnis zurückgeliefert. z.B.: <code>s= "WACHTURM"</code> ; <code>s=s.Remove(1, 4)</code> ; ergibt <code>s="WURM"</code>
<code>z=s.Insert(p, s1);</code>	z, s, s1: String; p: Integer;	Der <code>String s1</code> wird in <code>s</code> ab der Position <code>p</code> eingesetzt und ein neuer <code>String</code> erzeugt, z. B. <code>s="NORDWEST"</code> ; <code>s=s.Insert(4, "NORD")</code> ; ergibt <code>s="NORDNORDWEST"</code>
<code>r = s.Substring(p, n);</code>	r, s: String p, n: int	Aus dem <code>String s</code> werden ab der Position <code>p</code> <code>n</code> Zeichen (höchstens jedoch bis zum Ende des Strings) in die Zeichenkette <code>r</code> kopiert. Beispielsweise liefert <code>s="ABCDE"</code> ; <code>r=s.Substring(1, 3)</code> ; in <code>r</code> die Zeichenkette "BCD".
<code>r = s.Substring(p);</code>	r, s: String p: int	Aus dem <code>String s</code> werden ab der Position <code>p</code> alle Zeichen bis zum Ende von <code>s</code> in die Zeichenkette <code>r</code> kopiert.
<code>p=z.IndexOf(q);</code>	p: int q, z: String	Es wird untersucht an welcher Stelle der <code>String q</code> im Zielstring <code>z</code> erstmals vorkommt. Das Ergebnis ist die Nummer dieser Stelle. Kommt <code>q</code> in <code>z</code> nicht vor, ist <code>p=-1</code> . z. B. <code>z="Hallo Welt"</code> ; <code>z.IndexOf(" ")</code> ergibt 5
<code>p=z.IndexOf(q, r);</code>	q, r: int q, z: String	Im <code>String z</code> wird jetzt ab der Position <code>r</code> nach der Zeichenkette <code>q</code> gesucht. <code>p</code> liefert die Position.
<code>z = s.Replace(alt, neu);</code>	z, s: String alt, neu: String oder char	Durch die Methode <code>Replace</code> wird in der Zeichenkette <code>s</code> jedes Vorkommen von <code>alt</code> durch <code>neu</code> ersetzt und in einem neuen <code>String</code> gespeichert.

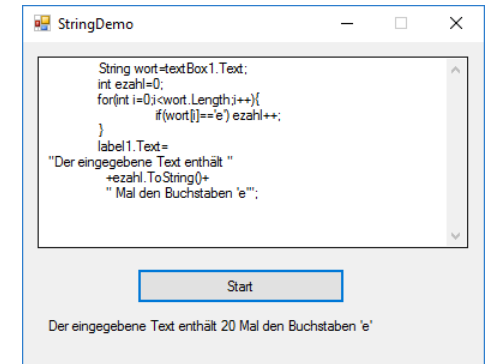
Eine Zahl wandelt man in einen `String` mit Hilfe der Methode `ToString()` der Zahl um, z. B. für `int x=3`; erhält man die dazu passende Zeichenkette durch `x.ToString()`; Einen `String` wandelt man in eine Zahl durch die Methode `Parse` um, wenn dies möglich ist. `Parse` wurde bereits besprochen.

Wichtig:

Bei allen Methoden, die den Inhalt einer Zeichenkette verändern, wird die geänderte Zeichenkette zurückgeliefert. Die ursprüngliche Zeichenkette bleibt unverändert.

Beispiel:

```
void Button1Click(object sender,
                  System.EventArgs e)
{
    String wort=textBox1.Text;
    int ezahl=0;
    for(int i=0;i<wort.Length;i++){
        if(wort[i]=='e') ezahl++;
    }
    label1.Text=
        "Der eingegebene Text enthält "
        +ezahl.ToString()+
        " Mal den Buchstaben 'e'";
}
```



Dieses kleine Beispielprogramm durchsucht eine eingegebene Zeichenkette Stelle für Stelle auf das Vorkommen des Zeichens "e",

Zwei Zeichenketten können auch mit `<` oder `>` verglichen werden. Dabei werden die Strings von links nach rechts zeichenweise verglichen. Beim ersten Auftreten von ungleichen Zeichen entscheidet die Lage (Nummer) im Unicode-Zeichensatz über kleiner oder größer. Strings kann man auch mit der Methode `Compare` vergleichen.

Beispiele: Es gilt: "balkon" > "balken" und "Georg" < "Hans"
Aber es gilt nicht: "Pascal" == "pascal"

Aufgaben:

- Ein Text von max. 35 Zeichen soll eingelesen werden und `g e s p e r r t` ausgegeben werden. Nach jedem Zeichen ist also ein Leerzeichen auszugeben.
- Ein C#-Programm soll bei Eingabe eines Wortes entscheiden, ob es sich dabei um ein so genanntes *Palindrom* handelt. Recherchiere gegebenenfalls im Internet.
- Ein eingegebenes Wort ist von hinten nach vorne gelesen in einer zweiten `String`-Variablen zu übergeben. Danach sind diese beiden Wörter aneinander zu hängen und das so entstandene neue Wort auszugeben.
- In einem Text sollen die Umlaute ä, ö, ü, Ä, Ö, Ü durch `ae`, `oe`, `ue`, `Ae`, `Oe`, `Ue` und `B` durch `ss` ausgedrückt werden. Das Programm soll zwei Lösungen enthalten. Eine mit Hilfe der Methoden von `Strings`, eine ohne diese Methoden.
- Nach Eingabe eines Textes soll der Text Wort für Wort untereinander in einer `Listbox` ausgedruckt werden. Das heißt in jeder Zeile soll nur ein Wort stehen. Verwende für das Zerlegen keine fertigen Methoden der Klasse `String`.
- Setze die Umwandlungen des Lieds „Drei Chinesen mit dem Kontrabass“ in C# um.