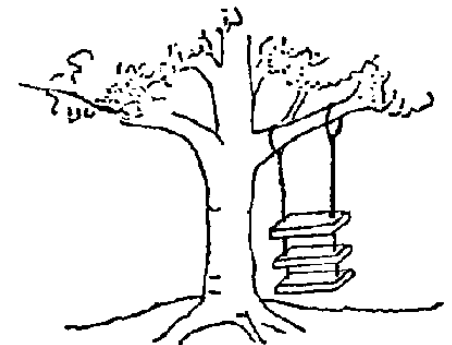


# Console-Ausgabe String-Funktionen StringBuilder Formen Beispiele

Software Entwicklung



# Übersicht

- Konsoleneingabe und Ausgabe
- Klasse String
- Klasse StringBuilder
- Formen in Dateien schreiben

# Konsolenausgabe

- Formatausdruck {}  
dient der eindeutigen Bestimmung des Elements
- ermöglicht auch eine Einflussnahme auf die Ausgabe
  - `Console.Write("{N,M} macht {N,M}.", t1, t2);`
- {N, M}
  - N ist ein nullbasierter Zähler.
  - M gibt die Breite der Ausgabe an.

# Beispiel Konsolenausgabe

- Wie lautet die Ausgabe:

- ```
int value = 10;  
Console.WriteLine("Ich kaufe {0,3} Semmel", value);  
Console.WriteLine("Ich kaufe {0,10} Semmel", value);
```

```
Ich kaufe  10 Semmel  
Ich kaufe           10 Semmel
```

- Die Breite darf auch eine negative Zahl sein.
- Die Ausgabe erfolgt dann linksbündig, daran schließen sich die Leerstellen an.

```
Ich kaufe 10  Semmel  
Ich kaufe 10           Semmel
```

# Konsolenausgabe

- Formatausdruck so spezifizieren, dass numerische Ausgabedaten eine bestimmte Formatierung annehmen:

```
// Syntax des Formatausdrucks  
{N [,M ][: Format]}
```

- `int value1 = 4711;`
- `Console.WriteLine("value={0:E}", value1);`
- `// Ausgabe: value=4,711000E+003`
  
- `Console.WriteLine("value={0:E2}", value1);`
- `// Ausgabe: value=4,71E+003`

# Formatangaben:

---

## Formatangabe Beschreibung

|   |                                                                           |
|---|---------------------------------------------------------------------------|
| C | Zeigt die Zahl im lokalen Währungsformat an.                              |
| D | Zeigt die Zahl als dezimalen Integer an.                                  |
| E | Zeigt die Zahl im wissenschaftlichen Format an (Exponentialschreibweise). |
| N | Zeigt eine numerische Zahl einschließlich Kommaseparatoren an.            |
| P | Zeigt die numerische Zahl als Prozentzahl an.                             |
| X | Die Anzeige erfolgt in Hexadezimalnotation.                               |

---

# Beispiele Konsolenausgabe

- `int value2 = 225;`
- `Console.WriteLine("value={0:X}", value2);`
- `Console.WriteLine("value={0:x}", value2);`  
    // Ausgabe: value=E1  
    // Ausgabe: value=e1
- `float value3 = 0.2512F;`
- `Console.WriteLine("value={0,10:G}", value3);`  
    // Ausgabe: value=      0,2512
- `Console.WriteLine("value={0:P4}", value3);`  
    // Ausgabe: value=25,1200%

# Escape Sequenzen

- Es soll folgende Zeichenkette ausgegeben werden:
  - Ich bin eine "Stringkonstante".
- Wie können „“ in eine Ausgabe eingefügt werden?
  - `Console.WriteLine("Ich bin eine \"Stringkonstante\".");`



# Escape-Zeichen

| Escape-Zeichen | Beschreibung                                                                      |
|----------------|-----------------------------------------------------------------------------------|
| \'             | Fügt ein Hochkomma in die Zeichenfolge ein.                                       |
| \"             | Fügt Anführungsstriche ein.                                                       |
| \\             | Fügt einen Backslash in die Zeichenfolge ein.                                     |
| \a             | Löst einen Alarmton aus.                                                          |
| \b             | Führt zum Löschen des vorhergehenden Zeichens.                                    |
| \n             | Löst einen Zeilenvorschub aus<br>(entspricht der Funktionalität der ENTER-Taste). |
| \r             | Führt zu einem Wagenrücklauf.                                                     |
| \t             | Führt auf dem Bildschirm zu einem Tabulatorsprung.                                |
| \v             | Fügt einen vertikalen Tabulator in eine Zeichenfolge ein.                         |

# Konsolenausgabe

- Wie kann folgende Ausgabe erzeugt werden:  
Hallo  
Welt
- `Console.WriteLine("Hallo\nWelt");`
- Wie kann folgende Ausgabe erzeugt werden:  
Hallo\nWelt
- `Console.WriteLine("Hallo\\nWelt");`
- Abschalten aller Interpretation als Escape-Sequenz für eine gegebene Zeichenfolge mit `>>@<<`
- `Console.Write(@"C#\nmacht\nSpaß.");`
- Ausgabe: `C#\nmacht\nSpaß.`

# Read vs. ReadLine

- Readline

- liest eine ganze Zeile und
- benutzt den Zeilenumbruch dazu, das Ende der Eingabe zu erkennen

- Read

- liest ein Zeichen ein und gibt den ASCII-Wert des Zeichens zurück.
- der Zeilenumbruch wird nicht aus dem Eingabestrom geholt, sondern verbleibt dort und wird so lange gepuffert, bis er von einer anderen Anweisung gelöscht wird. Das kann wiederum nur die Methode ReadLine sein

# ASCII Code Tabelle

| Dec | Bin       | Hex | Char  | Dec | Bin       | Hex | Char  | Dec | Bin       | Hex | Char | Dec | Bin       | Hex | Char  |
|-----|-----------|-----|-------|-----|-----------|-----|-------|-----|-----------|-----|------|-----|-----------|-----|-------|
| 0   | 0000 0000 | 00  | [NUL] | 32  | 0010 0000 | 20  | space | 64  | 0100 0000 | 40  | @    | 96  | 0110 0000 | 60  | `     |
| 1   | 0000 0001 | 01  | [SOH] | 33  | 0010 0001 | 21  | !     | 65  | 0100 0001 | 41  | A    | 97  | 0110 0001 | 61  | a     |
| 2   | 0000 0010 | 02  | [STX] | 34  | 0010 0010 | 22  | "     | 66  | 0100 0010 | 42  | B    | 98  | 0110 0010 | 62  | b     |
| 3   | 0000 0011 | 03  | [ETX] | 35  | 0010 0011 | 23  | #     | 67  | 0100 0011 | 43  | C    | 99  | 0110 0011 | 63  | c     |
| 4   | 0000 0100 | 04  | [EOT] | 36  | 0010 0100 | 24  | \$    | 68  | 0100 0100 | 44  | D    | 100 | 0110 0100 | 64  | d     |
| 5   | 0000 0101 | 05  | [ENQ] | 37  | 0010 0101 | 25  | %     | 69  | 0100 0101 | 45  | E    | 101 | 0110 0101 | 65  | e     |
| 6   | 0000 0110 | 06  | [ACK] | 38  | 0010 0110 | 26  | &     | 70  | 0100 0110 | 46  | F    | 102 | 0110 0110 | 66  | f     |
| 7   | 0000 0111 | 07  | [BEL] | 39  | 0010 0111 | 27  | '     | 71  | 0100 0111 | 47  | G    | 103 | 0110 0111 | 67  | g     |
| 8   | 0000 1000 | 08  | [BS]  | 40  | 0010 1000 | 28  | (     | 72  | 0100 1000 | 48  | H    | 104 | 0110 1000 | 68  | h     |
| 9   | 0000 1001 | 09  | [TAB] | 41  | 0010 1001 | 29  | )     | 73  | 0100 1001 | 49  | I    | 105 | 0110 1001 | 69  | i     |
| 10  | 0000 1010 | 0A  | [LF]  | 42  | 0010 1010 | 2A  | *     | 74  | 0100 1010 | 4A  | J    | 106 | 0110 1010 | 6A  | j     |
| 11  | 0000 1011 | 0B  | [VT]  | 43  | 0010 1011 | 2B  | +     | 75  | 0100 1011 | 4B  | K    | 107 | 0110 1011 | 6B  | k     |
| 12  | 0000 1100 | 0C  | [FF]  | 44  | 0010 1100 | 2C  | ,     | 76  | 0100 1100 | 4C  | L    | 108 | 0110 1100 | 6C  | l     |
| 13  | 0000 1101 | 0D  | [CR]  | 45  | 0010 1101 | 2D  | -     | 77  | 0100 1101 | 4D  | M    | 109 | 0110 1101 | 6D  | m     |
| 14  | 0000 1110 | 0E  | [SO]  | 46  | 0010 1110 | 2E  | .     | 78  | 0100 1110 | 4E  | N    | 110 | 0110 1110 | 6E  | n     |
| 15  | 0000 1111 | 0F  | [SI]  | 47  | 0010 1111 | 2F  | /     | 79  | 0100 1111 | 4F  | O    | 111 | 0110 1111 | 6F  | o     |
| 16  | 0001 0000 | 10  | [DLE] | 48  | 0011 0000 | 30  | 0     | 80  | 0101 0000 | 50  | P    | 112 | 0111 0000 | 70  | p     |
| 17  | 0001 0001 | 11  | [DC1] | 49  | 0011 0001 | 31  | 1     | 81  | 0101 0001 | 51  | Q    | 113 | 0111 0001 | 71  | q     |
| 18  | 0001 0010 | 12  | [DC2] | 50  | 0011 0010 | 32  | 2     | 82  | 0101 0010 | 52  | R    | 114 | 0111 0010 | 72  | r     |
| 19  | 0001 0011 | 13  | [DC3] | 51  | 0011 0011 | 33  | 3     | 83  | 0101 0011 | 53  | S    | 115 | 0111 0011 | 73  | s     |
| 20  | 0001 0100 | 14  | [DC4] | 52  | 0011 0100 | 34  | 4     | 84  | 0101 0100 | 54  | T    | 116 | 0111 0100 | 74  | t     |
| 21  | 0001 0101 | 15  | [NAK] | 53  | 0011 0101 | 35  | 5     | 85  | 0101 0101 | 55  | U    | 117 | 0111 0101 | 75  | u     |
| 22  | 0001 0110 | 16  | [SYN] | 54  | 0011 0110 | 36  | 6     | 86  | 0101 0110 | 56  | V    | 118 | 0111 0110 | 76  | v     |
| 23  | 0001 0111 | 17  | [ETB] | 55  | 0011 0111 | 37  | 7     | 87  | 0101 0111 | 57  | W    | 119 | 0111 0111 | 77  | w     |
| 24  | 0001 1000 | 18  | [CAN] | 56  | 0011 1000 | 38  | 8     | 88  | 0101 1000 | 58  | X    | 120 | 0111 1000 | 78  | x     |
| 25  | 0001 1001 | 19  | [EM]  | 57  | 0011 1001 | 39  | 9     | 89  | 0101 1001 | 59  | Y    | 121 | 0111 1001 | 79  | y     |
| 26  | 0001 1010 | 1A  | [SUB] | 58  | 0011 1010 | 3A  | :     | 90  | 0101 1010 | 5A  | Z    | 122 | 0111 1010 | 7A  | z     |
| 27  | 0001 1011 | 1B  | [ESC] | 59  | 0011 1011 | 3B  | ;     | 91  | 0101 1011 | 5B  | [    | 123 | 0111 1011 | 7B  | {     |
| 28  | 0001 1100 | 1C  | [FS]  | 60  | 0011 1100 | 3C  | <     | 92  | 0101 1100 | 5C  | \    | 124 | 0111 1100 | 7C  |       |
| 29  | 0001 1101 | 1D  | [GS]  | 61  | 0011 1101 | 3D  | =     | 93  | 0101 1101 | 5D  | ]    | 125 | 0111 1101 | 7D  | }     |
| 30  | 0001 1110 | 1E  | [RS]  | 62  | 0011 1110 | 3E  | >     | 94  | 0101 1110 | 5E  | ^    | 126 | 0111 1110 | 7E  | ~     |
| 31  | 0001 1111 | 1F  | [US]  | 63  | 0011 1111 | 3F  | ?     | 95  | 0101 1111 | 5F  | _    | 127 | 0111 1111 | 7F  | [DEL] |

# Farben in der Konsole

- Vordergrundfarbe setzen:
  - `Console.ForegroundColor = ...`
- Hintergrundfarbe setzen:
  - `Console.BackgroundColor = ...`
- Farben wählen
  - `ConsoleColor.Black;`
  - `ConsoleColor.White;`

# String & Char

- Jede Zeichenkette besteht aus Buchstaben
- Jeder string besteht aus chars...
- Mit [...] kann man auf Buchstaben zugreifen

```
0: H  
1: a  
2: l  
3: l  
4: o
```

```
static void Main(string[] args) {  
    string s = "Hallo";  
    for (int i = 0; i < s.Length; i++) {  
        Console.WriteLine($"{i}: {s[i]}");  
    }  
}
```

# String und Char

- Umgekehrte Reihenfolge

```
static void Main(string[] args) {  
    string s = "Hallo";  
    for (int i = s.Length-1; i >= 0; i--) {  
        Console.WriteLine($"{i}: {s[i]}");  
    }  
}
```

4: o  
3: l  
2: l  
1: a  
0: H

- Ausgabe eines Char-Arrays

```
char[] cArray = { 'H', 'a', 'l', 'l', 'o' };  
for (int i = 0; i < cArray.Length; i++) {  
    Console.WriteLine($"{i}: {cArray[i]}");  
}
```

0: H  
1: a  
2: l  
3: l  
4: o

# String erzeugen mit Chars

- Aus Char Array

```
Console.WriteLine("CharArray an String übergeben");  
string strText = new string(cArray);  
Console.WriteLine(strText);
```

```
CharArray an String übergeben  
Hallo
```

- Aus einzelnen Chars

```
Console.WriteLine("\nSterne erzeugen");  
string strStar = new string('*', 10);  
Console.WriteLine(strStar);  
Console.WriteLine("\nUnterstiche erzeugen");  
string strUnderscore = new string('_', strStar.Length);  
Console.WriteLine(strUnderscore);
```

```
Sterne erzeugen  
*****  
  
Unterstiche erzeugen  
_____
```



# Länge einer Zeichenkette

- Mit dem Property Length

```
string s = "hallo welt";  
int len = s.Length;  
Console.WriteLine("{0} ist {1} Zeichen lang", s, len);
```

```
//Ausgabe Zeichenweise  
for (int i = 0; i < s.Length; i++) {  
    Console.Write("{0} ", s[i]);  
}
```

# Zeichenketten

- Zum Vergleichen:
  - Equals
  - Compare
  - CompareTo
- Zum Verändern
  - Insert
  - Replace
  - Remove
  - PadLeft & PadRight
  - Substring
- Eigenschaft
  - Length
- Zum Suchen
  - IndexOf
  - LastIndexOf
  - Contains
  - StartsWith
  - EndsWith

# Zeichenketten vergleichen mit Equals

- „Equals“ kontrolliert die Gleichheit byteweise

```
string text = "Hallo";  
string text1 = text;  
string text2 = "Hallo";
```

```
0: Zeichenketten sind gleich  
1: Zeichenketten sind gleich  
2: Zeichenketten sind gleich
```

```
if (text == text1 && text == text2) {  
    Console.WriteLine("0: Zeichenketten sind gleich");  
}  
  
//Equals  
if (text.Equals(text1))  
    Console.WriteLine("1: Zeichenketten sind gleich");  
if (text.Equals(text2))  
    Console.WriteLine("2: Zeichenketten sind gleich");
```

# Zeichenketten vergleichen mit CompareTo

- Inhalt der Zeichenkette wird zeichenweise verglichen mit der Instanzmethode

```
//Compare  
int i = text.CompareTo(text1);  
Console.WriteLine("3: Ausgabe {0}", i);  
i = text.CompareTo(text2);  
Console.WriteLine("4: Ausgabe {0}", i);
```

```
3: Ausgabe 0  
4: Ausgabe 0
```

# Vergleichen mit String.Compare

- Zeichenketten vergleichen mit der Klassenmethode:

```
//Compare
string name1 = "Anton";
string name2 = "Kurt";
string name3 = "Xaver";
int j = String.Compare(name1, name2);
Console.WriteLine($"{name1} vs {name2} = {j}");
j = String.Compare(name2, name3);
Console.WriteLine($"{name2} vs {name3} = {j}");
j = String.Compare(name3, name1);
Console.WriteLine($"{name3} vs {name1} = {j}");
```

```
Anton vs Kurt = -1
Kurt vs Xaver = -1
Xaver vs Anton = 1
```

# Vergleichen mit String.Compare

- Zeichenketten vergleichen mit der Klassenmethode:

```
text = "Hallo";  
text2 = "hallo";  
//Compare  
i = String.Compare(text, text2);  
Console.WriteLine("5: Ausgabe {0}", i);
```

5: Ausgabe 1

| Rückgabewert | Beschreibung                                                     |
|--------------|------------------------------------------------------------------|
| < 0          | Der String des ersten Arguments ist kleiner als der des zweiten. |
| 0            | Beide Strings sind gleich.                                       |
| > 0          | Der String des ersten Arguments ist größer als der des zweiten.  |

# Alles Groß vs Alles Klein

mit `variable.ToUpper()`  
und `variable.ToLower()`

```
text = "Visual Studio .NET";  
text = text.ToUpper(); // VISUAL STUDIO .NET  
Console.WriteLine(text);
```

```
string ss = "Hello, World!";  
string tt = ss.
```

- Substring
- ToCharArray
- ToLower
- ToLowerInvariant
- ToString
- ToUpper**
- ToUpperInvariant
- Trim
- TrimEnd

`string string.ToUpper(System.Globalization.CultureInfo culture)` (+ 1 overload(s))  
Returns a copy of this string converted to uppercase, using the casing rules of the specified culture.

Exceptions:  
`System.ArgumentNullException`

# Zeichenfolge ändern

- ToUpper & ToLower

wandeln alle Zeichen einer Zeichenfolge entweder

- in Großbuchstaben (ToUpper) oder
- in Kleinbuchstaben (ToLower) um

```
text = "HaLlO Das Ist Bunt Gemischter Text";  
Console.WriteLine("Groß:\t" + text.ToUpper());  
Console.WriteLine("Klein:\t" + text.ToLower());
```

```
Groß:  HALLO DAS IST BUNT GEMISCHTER TEXT  
Klein: hallo das ist bunt gemischter text
```



# Zeichenfolge ändern

- Insert -> Einfügen

```
public string Insert(int, string);
```

- Replace -> Ersetzen

```
public string Replace(char, char);  
public string Replace(string, string);
```

- Remove -> Entfernen

```
public string Remove( int startIndex, int count);
```

# Strings einfügen & Ersetzen

- Insert - Einfügen

```
string text = "Hallo";  
string newText = text.Insert(1, "aaaaa");  
Console.WriteLine($"{text} mit ein paar a mehr: {newText}");
```

- Replace - Ersetzen

```
newText = text.Replace('a', 'e');  
Console.WriteLine($"{text} wird: {newText}");  
newText = text.Replace("allo", "ellllllo");  
Console.WriteLine($"{text} wird: {newText}");  
  
text = "Mein Hallo ist ein Ersatz für Grüß Dich";  
text = text.ToLower();  
newText = text.Replace('i', '!');  
newText = newText.Replace('e', '3');  
Console.WriteLine($"{text} \n\twird: {newText}");
```

```
Hallo mit ein paar a mehr: Haaaaaallos  
Hallo wird: Hello  
Hallo wird: Helllllllo
```

```
mein hallo ist ein ersatz für grüß dich  
wird: m3!n hallo !st 3!n 3rsatz für grüß d!ch
```

# Entfernen

- Remove - Entfernen

```
newText = text.Remove(2);  
Console.WriteLine($"{text} wird: {newText}");  
Console.WriteLine("Entfernt ab der Stelle 2 eine Anzahl von 2 Buchstaben");  
newText = text.Remove(2, 2);  
Console.WriteLine($"{text} wird: {newText}");
```

```
Entfernt ab der Stelle 2 alle Buchstaben:  
Hallo wird: Ha  
Entfernt ab der Stelle 2 eine Anzahl von 2 Buchstaben  
Hallo wird: Hao
```

```
text = "Mein neuer Text";  
Console.WriteLine("\nLänge: " + text.Length);  
newText = text.Remove(7, 1);  
Console.WriteLine($"0: {text} wird: {newText}");  
newText = text.Remove(2, 5);  
Console.WriteLine($"1: {text} wird: {newText}");  
newText = text.Remove(10, 3);  
Console.WriteLine($"2: {text} wird: {newText}");
```

```
Länge: 15  
0: Mein neuer Text wird: Mein neer Text  
1: Mein neuer Text wird: Meuer Text  
2: Mein neuer Text wird: Mein neuerxt
```

# Position finden

- Von einem Buchstaben:

```
string text = "Hallo";  
char c = 'a';  
int index = text.IndexOf(c);  
Console.WriteLine($"Index von {c} in {text} ist {index}");
```

Index von a in Hallo ist 1

- Von einer Zeichenkette (Substring):

```
string substr = "al";  
index = text.IndexOf(substr);  
Console.WriteLine($"Index von {substr} in {text} ist {index}");
```

Index von al in Hallo ist 1

# Alle Positionen finden

```
string text = "Hallo";  
char c = 'l';  
int index = text.IndexOf(c);  
Console.WriteLine($"{text} hat '{c}' an Stelle {index}");  
c = 'z';  
index = text.IndexOf(c);  
Console.WriteLine($"{text} hat '{c}' an Stelle {index}");
```

```
Hallo hat 'l' an Stelle 2  
Hallo hat 'z' an Stelle -1
```

```
HalloHallooo hat an Stelle 4,9,10,11, den Buchstaben o
```

```
text = text + "Hallooo";  
c = 'o';  
index = -1;  
Console.Write($"{text} hat an Stelle ");  
do {  
    index++;  
    index = text.IndexOf(c, index);  
    if (index != -1)  
        Console.Write(index + ",");  
} while (index != -1);  
Console.WriteLine($" den Buchstaben {c}");
```

# Insert - Einfügen in eine Zeichenkette

- Zeichenfolge ab einer bestimmten Position zu erweitern, ohne dabei aus dem Original Zeichen durch Überschreiben zu löschen
- **Syntax:**

```
public string Insert(int, string);  
string text1 = "C# ist spitze.";   
text1 = text1.Insert(text1.IndexOf("spitze"), "absolut ");  
Console.WriteLine(text1);
```
- Ausgabe? `C# ist absolut spitze.`

# Replace & Remove

## Ersetzen und Entfernen einer Zeichenkette

- Replace ersetzt eine Zeichenkette oder einzelne Zeichen:

```
public string Replace(char, char);  
public string Replace(string, string);
```

- Remove löscht eine Anzahl von Zeichen ab einer spezifizierten Position aus der Zeichenfolge.
- `public string Remove( int startIndex, int count);`

```
string text1 = "C# ist spitze.";
text1 = text1.Insert(text1.IndexOf("spitze"), "absolut ");
Console.WriteLine(text1);
text1 = text1.Replace("spitze", "genial");
Console.WriteLine(text1);
text1 = text1.Remove(text1.IndexOf("absolut"), "absolut".Length+1);
Console.WriteLine(text1);
```

Ausgabe?

```
C# ist absolut spitze.  
C# ist absolut genial.  
C# ist genial.
```

# Finde einen Buchstaben

```
static void FindLetter() {  
    string word = Console.ReadLine();  
    char letter = char.Parse(Console.ReadLine());  
    for (int i = 0; i < word.Length; i++) {  
        if (word[i] == letter)  
            Console.WriteLine("{0} an der" +  
                               " Stelle {1} gefunden", letter, i);  
    }  
    Console.WriteLine(word);  
    Console.WriteLine(letter);  
}
```



# Beispiel Ersetzen mit Replace

- Zeichenkette einlesen
- Buchstaben 1 und Buchstaben 2 einlesen
- Buchstaben 1 durch Buchstaben 2 ersetzen:

```
Console.WriteLine("Bitte geben sie ein Wort ein:");
string word = Console.ReadLine();
Console.WriteLine("Bitte geben sie einen Buchstaben ein der ersetzt werden soll:");
char letter = char.Parse(Console.ReadLine());
Console.WriteLine("Bitte geben sie einen Buchstaben ein mit welchem er ersetzt werden soll:");
char replaceletter = char.Parse(Console.ReadLine());
word = word.Replace(letter, replaceletter);
Console.WriteLine(word);
Console.WriteLine("\n\nVersion 2 selbst ersetzen");
Console.WriteLine("Bitte geben sie ein Wort ein:");
string word1 = Console.ReadLine();
```

Erstelle das Beispiel mit Remove und Insert - statt Replace, IndexOf darf benutzt werden.

# Suchen in Zeichenketten

- Suche nach Buchstaben und Substrings:
  - Feststellen ob die Zeichenkette mit einem
    - bestimmten Buchstaben/Wort (Substring) anfängt.
    - Bestimmten Buchstaben/Wort (Substring) aufhört.

```
string text = "Hallo";  
if (text.StartsWith('H'))  
    Console.WriteLine("Startet mit \"H\"");  
if (text.EndsWith('o'))  
    Console.WriteLine("Endet mit \"o\"");  
if (text.StartsWith("Hal"))  
    Console.WriteLine("Startet mit \"Hal\"");  
if (text.EndsWith("llo"))  
    Console.WriteLine("Endet mit \"llo\"");
```

```
Startet mit "H"  
Endet mit "o"  
Startet mit "Hal"  
Endet mit "llo"
```

# Position eines Teilstrings finden

- IndexOf liefert die Position des gesuchten Teilstrings:

```
string text = "Da wird der Hund in der Pfanne verrückt.";
Console.WriteLine(text);
int position = -1;
do {
    position++;
    position = text.IndexOf("der", position);
    if(position == -1)
        Console.WriteLine("Ende des Strings erreicht.");
    else
        Console.WriteLine("Vorkommen an Position {0}", position);
} while(!(position == -1));
```

- ```
Da wird der Hund in der Pfanne verrückt.
```
- Vorkommen an Position 8  
Vorkommen an Position 20  
Ende des Strings erreicht. \_

# Zeichen hinzufügen

- zu einer Zeichenkette
  - Links oder Rechts

```
string text = "hallo";  
Console.WriteLine(text.PadLeft(8));  
Console.Write(text.PadRight(8));  
Console.WriteLine(".");
```

```
string newText = text.PadLeft(8, '*');  
Console.WriteLine(newText);  
newText = newText.PadRight(newText.Length + 3, '*');  
Console.WriteLine(newText);
```

```
hallo  
hallo .  
***hallo  
***hallo***
```

# Padding: PadLeft - PadRight

```
string text1 = "C# macht Spass";  
Console.WriteLine(text1.PadLeft(text1.Length + 3));  
text1 = text1.PadLeft(text1.Length + 3, '*');  
Console.WriteLine(text1);  
text1 = text1.PadRight(text1.Length + 3, '*');  
Console.WriteLine(text1);  
string text2 = "";  
Console.WriteLine(text2.PadLeft(10, '-'));  
Console.WriteLine(text2);
```

- Wie lautet die Ausgabe?

```
      C# macht Spass  
***C# macht Spass  
***C# macht Spass***  
-----
```

```
- . . . . .
```

# Zeichenkette mit Zeichen auffüllen

- Mit den Methoden `PadLeft` und der Angabe einer Zahl wird die Zeichenkette mit Leerzeichen gefüllt.
- Alternativ kann ein beliebiges Zeichen zum Auffüllen verwendet werden.

- **Syntax:**

```
public string PadLeft(int);  
public string PadLeft(int, char);  
public string PadRight(int);  
public string PadRight(int, char);
```

# Teilstring kopieren

- Zum kopieren eines Teils der Zeichenkette wird der Startindex und die Anzahl der zu kopierenden Zeichen benötigt:
- **Syntax:**

```
String Substring(int startIndex, int length)
//Teilstring
string text3 = "Projektmappen-Explorer";
string teilString = text3.Substring(0, 7);
Console.WriteLine(teilString);
teilString = text3.Substring(text3.Length - 8, 8);
Console.WriteLine(teilString);
```

- Wie lautet die Ausgabe? Projekt Explorer

# Teilstring rauskopieren

- Ab einer Indexposition kopieren, bis ans Ende der Zeichenkette

```
string text = "hallo welt";  
string sub = text.Substring(5);  
Console.WriteLine($"Substring von {text} ab Stelle 5 ist");  
Console.WriteLine(sub);
```

- Ab einer Indexposition kopieren, mit einer konkreten Anzahl von Stellen, die kopiert werden sollen

```
sub = text.Substring(3, 4);  
Console.WriteLine($"Substring (ab 3 mit 4 Zeichen) " +  
    $"von {text} ist \n{sub}");
```

- ```
Substring von hallo welt ab Stelle 5 ist  
welt  
Substring (ab 3 mit 4 Zeichen) von hallo welt ist  
lo w
```





# StringBuilder

Zeichenketten (String) können nicht verändert, nur verworfen und neu erstellt werden - StringBuilder ermöglicht auch ein Verändern einer Zeichenkette

# String

- Objekte der Klasse **String** sind **unveränderlich**,
- bei jeder Änderung im Speicher wird ein neues Objekt angelegt
- sogar unabhängig davon, ob sich die Länge der Zeichenkette ändert
- Führen Sie viele manipulierende Operationen aus, hat das Einbußen der Systemleistung zur Folge.

# StringBuilder

- Zeichenfolge häufig verändert
  - vorhandenen Ressourcen möglicherweise knapp, mit der Konsequenz der Performanceeinbuße
- Falls eine Zeichenfolge oft geändert wird:  
Klasse **StringBuilder** nutzen,  
die die beschriebenen Nachteile nicht hat.
- **Hinweis:**
- Fähigkeiten einer StringBuilder-Zeichenfolge deutlich eingeschränkter als die einer string-Zeichenfolge

# Eigenschaften von String

- Länge eines Strings

```
string text = "Hallo,,;  
Console.WriteLine("Länge: " + text.Length);  
//Ausgabe: „Länge: 5“
```

- Einzelne Zeichen eines Strings abfragen

```
string text = "HALLO";  
char newChar = text[2];
```

- Die char-Variable enthält damit das Zeichen »L«.

# Schreibrechte String vs StringBuilder

- Keine Schreibrechte auf String
- Nutze den StringBuilder

```
string s = "Hallo";  
char c = s[1];  
Console.WriteLine(s[4]);  
string newS = "Halllo";  
StringBuilder sb = new StringBuilder(newS);  
newS[4] = 'o'; //Fehler - keine Schreibrechte  
sb[4] = 'o';   //Lösung - StringBuilder Benutzen
```

# Ersetzen mit StringBuilder

```
//StringBuilder
Console.WriteLine("Bitte geben sie ein Wort ein:");
string word1 = Console.ReadLine();
Console.WriteLine("Bitte geben sie einen Buchstaben ein der ersetzt werden soll:");
char searchletter2 = char.Parse(Console.ReadLine());
Console.WriteLine("Bitte geben sie einen Buchstaben ein mit welchem er ersetzt werden soll:");
char replaceletter2 = char.Parse(Console.ReadLine());
StringBuilder sb = new StringBuilder(word1);
for (int i = 0; i < word1.Length; i++) {
    if (word1[i] == searchletter2) {
        sb[i] = replaceletter2;
        //word1[i] = replaceletter2; //Fehler - keine Schreibrechte
    }
}
string word3 = sb.ToString();
Console.WriteLine(sb.ToString());
Console.WriteLine(sb);
Console.WriteLine(word3);
```

# Wie groß ist der Zeitunterschied?

```
static void testString()
{
    Stopwatch watch = new Stopwatch();
    string text = "";
    watch.Start();
    for (int i = 0; i < 50000; i++)
        text += "x";
    watch.Stop();
    Console.WriteLine("Zeit String: {0}", watch.ElapsedMilliseconds);
}

static void testStringBuilder()
{
    Stopwatch watch = new Stopwatch();
    StringBuilder str = new StringBuilder();
    watch.Start();
    for (int i = 0; i < 50000; i++)
        str = str.Append("x");
    watch.Stop();
    Console.WriteLine("Zeit StringBuilder: {0}", watch.ElapsedMilliseconds);
}
```

|                       |
|-----------------------|
| Zeit String: 950      |
| Zeit StringBuilder: 0 |

# Eigenschaften der Klasse StringBuilder

- **Chars**
- Liefert das Zeichen an einer genau spezifizierten Position aus der Zeichenfolge. Diese Eigenschaft ist der Indexer der Klasse.
- *Hinweis:*
  - Die Klasse String bietet nur die Möglichkeit zum Abfragen, nicht zum Setzen eines Chars an einer bestimmten Stelle.
- **Length**
- Liefert die Länge der Zeichenfolge.



# Länge abfragen

## Zeichen verändern / abfragen

- Im StringBuilder können Zeichen abgefragt werden
- Zeichen an spezifischen Stellen überschrieben werden.
- Der Index beginnt bei 0 zu zählen.

```
StringBuilder builder = new StringBuilder("Freitagabend");  
char c = builder[4];  
Console.WriteLine(c);  
builder[3] = '!';  
Console.WriteLine("{1}\nLänge: {0}", builder.Length, builder);
```

- Wie lautet die Ausgabe?

```
t  
Fre!tagabend  
Länge: 12
```

# Methoden der Klasse StringBuilder

| Method       | Eigenschaft                                                                     |
|--------------|---------------------------------------------------------------------------------|
| Append       | Hängt eine Zeichenfolge an eine bestehende StringBuilder-Instanz an.            |
| AppendFormat | Fügt der StringBuilder-Instanz eine Zeichenfolge mit Formatangaben an.          |
| AppendLine   | Eine Zeile hinzufügen                                                           |
| CopyTo       | Kopiert einen Teil des Objekts in ein char-Array.                               |
| Insert       | Fügt an einer spezifizierten Position eine Zeichenfolge ein.                    |
| Remove       | Löscht aus einer Zeichenfolge ab einer bestimmten Position eine Zeichensequenz. |
| Replace      | Ersetzt in der gesamten Zeichenfolge ein Zeichen durch ein anderes.             |

# Bearbeiten eines StringBuilder-Objekts

- Append      anhängen/einfügen einer Zeichenkette
- Insert      hinzufügen einer Zeichenkette
- Remove      entfernen einer Zeichenkette
- Replace      ersetzen einer Zeichenkette

```
StringBuilder builder2 = new StringBuilder();  
builder2 = builder2.Append("fällt Schnee");  
Console.WriteLine(builder2);  
builder2 = builder2.Insert(0, "Im Winter ");  
Console.WriteLine(builder2);  
builder2 = builder2.Remove(3, 2);  
Console.WriteLine(builder2);  
builder2 = builder2.Replace("nter", "Sommer");  
Console.WriteLine(builder2);
```

- Ausgabe?

```
fällt Schnee  
Im Winter fällt Schnee  
Im nter fällt Schnee  
Im Sommer fällt Schnee
```

# Formen darstellen

## Schreibe diese Formen in eine Datei

Quadrat ausgeben

Dreieck ausgeben

Einfacher Tannenbaum

Gestuftes Tannenbaum

Kreis/e ausgeben -> Schneemann ausgeben

Stern ausgeben

Form deiner Wahl - Angabe & Lösung erstellen

```
  *
 ***
*****
 ***
*****
*****
*****
*****
*****
#####
###
###
```

# Quadrat ausgeben

- Erstelle eine Consolenapp und lese eine Zahl zwischen 4 und ... (max) ein, und ein Zeichen für die Darstellung. Erzeuge...
  - Version 1
    - ein Quadrat in der Console
  - Version 2
    - Ein Quadrat in der Console mit ein Zeichen als Kantenbreite in der Mitte
  - Version 3
    - Ein Quadrat mit 3 Zeichen als Kantenbreite in der Mitte mindestens einem Leerzeichen (min = 1, max = ...)
- Ändere dein Programm, dass das darstellende Zeichen veränderbar oder einlesbar ist.

```
#####
#####
#####
#####
#####
#####
```

```
#####
#       #
#       #
#       #
#       #
#####
```

```
#####
#####
#####
###    ###
###    ###
###    ###
#####
#####
#####
#####
```



- Erstelle einen Tannenbaum in variabler Höhe, entscheide selbst deine Baumarten. Es ist erlaubt beide Arten von Baum zu implementieren.

- 
- nmhöhe

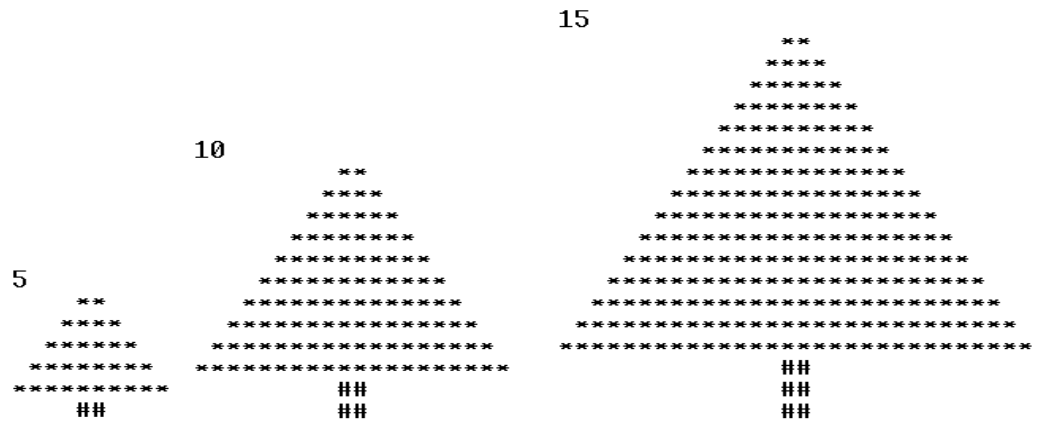
# Hinweis: String.PadRight / PadLeft

- Zum Erstellen der Zeichenkette darf die Funktion String.PadRight oder String.PadLeft verwendet werden.
- Du kannst deinen Tannenbaum auch mit unterschiedlichen Zeichen erstellen lassen.
  - `string str = "forty-two";`
  - `char pad = '.';`
  - `Console.WriteLine(str.PadRight(15, pad));`  
`// Displays "forty-two.....".`
  - `Console.WriteLine(str.PadRight(2, pad));`  
`// Displays "forty-two".`



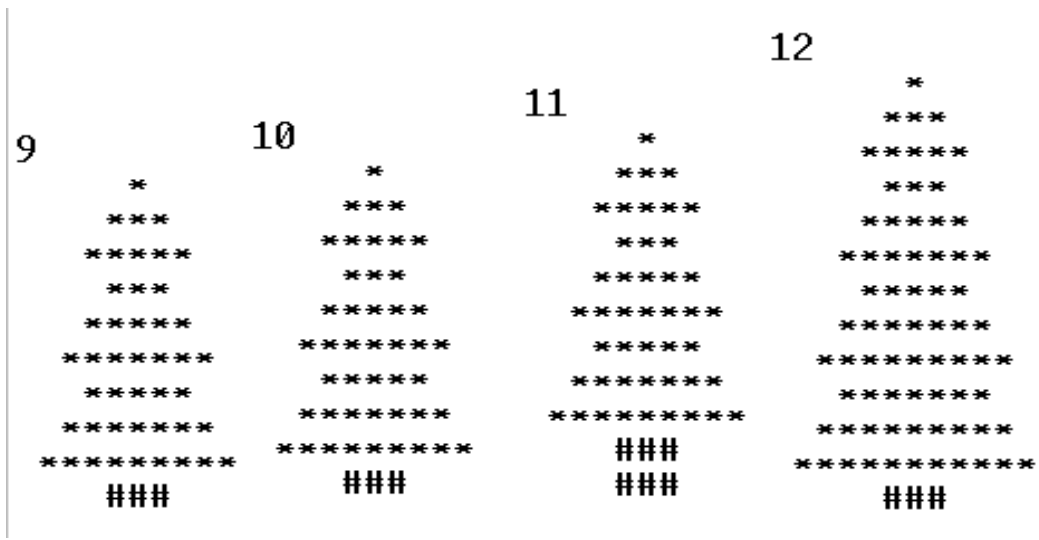
# Variante 1

- Ein möglicher Lösungsansatz ist die Höhe für den Baum ohne Stamm zu verwenden und in Relation dazu 1/3 zusätzlich für den Stamm zu ergänzen. Andere Lösungsansätze sind ebenfalls erlaubt:



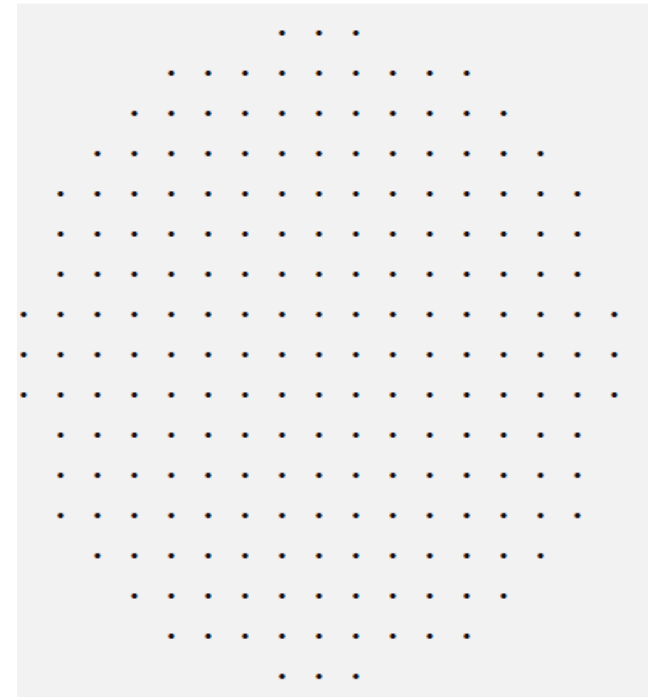
# Variante 2

- Ein möglicher Lösungsansatz ist es die Baumstammhöhe zum Ausgleichen der Gesamthöhe zu verwenden. Andere Lösungsansätze sind ebenfalls erlaubt:



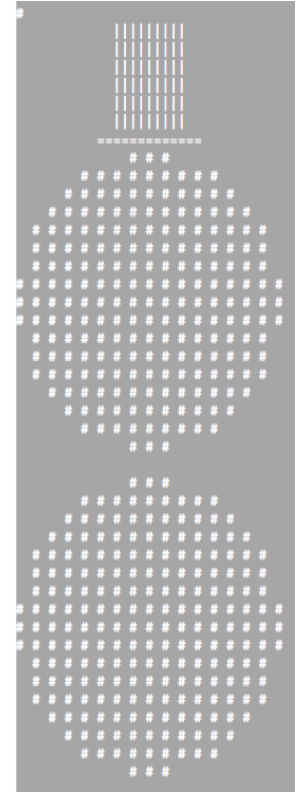
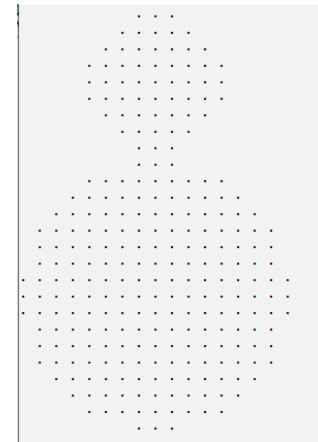
# Kreis ausgeben

Überlege dir die Ausgabe für  
einen Kreis beliebiger Größe  
mit einem beliebigen Zeichen



# Schneemann

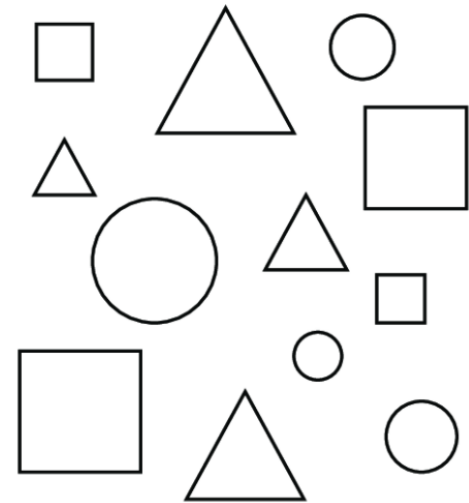
Erstelle die Ausgabe eines Schneemanns mit 2-3 Kreisen übereinander. Gestalte nach eigenem Ermessen.



```
  ##  
  ####  
#####  
#####  
#####  
#####  
  ##  ##  
  #    #
```

# Stern

Erzeuge eine Ausgabe, die einen Stern ausgibt. Erstelle dies zuerst mit einer fixen Größe. Überlege dir ob eine Ausgabe einer variablen Größe auch lösbar ist.



## Eigene Form

Erfinde eine Form deiner Wahl, die mit beliebiger Größe und beliebigen Zeichen darstellbar ist. Erstelle eine Angabe und eine Musterlösung - dieses Beispiel sollte einer deiner Mitschüler laut deiner Angabe lösen können.

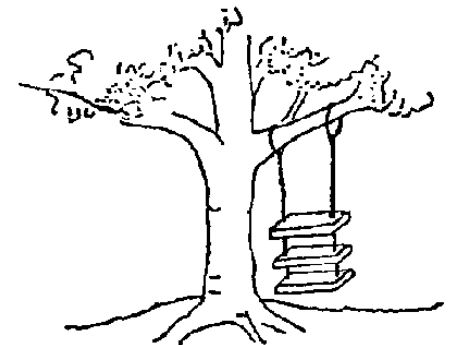
# Mögliche unterschiedliche Lösungsansätze für die Console

Quadrat (Version 1-3)

Dreieck (Version 1-3)

Tannenbaum (Version 1-2)

Kreis



# Ausgabe der Quadrate

- Version 1
- Version 2
- Version 3

```
# # # # # #  
# # # # # #  
# # # # # #  
# # # # # #  
# # # # # #  
# # # # # #
```

```
# # # # # #  
#           #  
#           #  
#           #  
#           #  
#           #  
# # # # # #
```

```
# # # # # # # #  
# # # # # # # #  
# # # # # # # #  
# # #       # # #  
# # #       # # #  
# # #       # # #  
# # # # # # # #  
# # # # # # # #  
# # # # # # # #
```



# Version 1 - Quadrat

- 2 verschachtelte Schleifen
  - In der inneren die Ausgabe des Zeichens
  - In der äußeren das Enter nach einer Zeile

```
int a = 6; // Int32.Parse(Console.ReadLine());
char c = '#'; // Char.Parse(Console.ReadLine());
for (int i = 0; i < a; i++) {
    for (int j = 0; j < a; j++) {
        Console.Write( c);
    }
    Console.WriteLine();
}
```

```
#####
#####
#####
#####
#####
#####
```

# Version 1 - Quadrat

- Mit PadRight

```
string a;  
Console.Write("Bitte die Größe: ");  
int b = Convert.ToInt32(Console.ReadLine());  
Console.Write("Bitte beliebes Zeichen auswählen: ");  
char c = Convert.ToChar(Console.ReadLine());  
for (int i = 0; i < b; i++) {  
    a = "";  
    a = a.PadRight(b, c);  
    Console.WriteLine(a);  
}
```

```
#####  
#####  
#####  
#####  
#####  
#####
```

# Quadrat Version 1

- Mit StringBuilder

```
int a = 6; // Int32.Parse(Console.ReadLine());
char c = '#'; // Char.Parse(Console.ReadLine());
StringBuilder s = new StringBuilder();
for (int i = 0; i < a; i++) {
    for (int j = 0; j < a; j++) {
        s = s.Append(c);
    }
    s = s.AppendLine();
}
Console.WriteLine(s);
```

```
#####
#####
#####
#####
#####
#####
```

# Version 2 - Quadrat

- Console Write  
Console Write Line

```
int amount = 6; // Int32.Parse(Console.ReadLine());
char c = '#'; // Char.Parse(Console.ReadLine());

//Ausgabe der ersten Zeile mit einem Zeichen amount mal
for (int j = 0; j < amount; j++) {
    Console.Write(c);
}
Console.WriteLine();

//4 x (amount minus erste und letzte Zeile) Mal die Zeile
for (int i = 0; i < amount - 2; i++) {

    //erstes Zeichen am Zeilenbeginn
    Console.Write(c);
    //amount minus erstes und letztes Zeichen mal ein Leerzeichen
    for (int i1 = 0; i1 < amount - 2; i1++) {
        Console.Write(" ");
    }
    //letztes Zeichen und Zeilenende (Enter)
    Console.WriteLine(c);
}

// Ausgabe der ersten Zeile mit einem Zeichen amount mal
for (int i = 0; i < amount; i++) {
    Console.Write(c);
}
Console.WriteLine();
```



```
#####
#      #
#      #
#      #
#      #
#####
```

# Version 2 - Quadrat

- 2 Lösung mit StringBuilder

```
int a = 6; // Int32.Parse(Console.ReadLine());
char c = '#'; // Char.Parse(Console.ReadLine());
StringBuilder s = new StringBuilder();
for (int i = 0; i < a; i++) {
    s = s.Append(c);
}
s = s.Append("\n");
for (int i = 0; i < a - 2; i++) {
    s = s.Append(c);

    for (int i1 = 0; i1 < a - 2; i1++) {
        s = s.Append(" ");
    }
    s = s.Append(c);
    s = s.Append("\n");
}
for (int i = 0; i < a; i++) {
    s = s.Append(c);
}
Console.WriteLine(s);
```



```
#####
#       #
#       #
#       #
#       #
#####
```

# Version 2 - Quadrat

- Eine Lösung mit Pad-Right

```
string a;  
Console.Write("Bitte die Größe: ");  
int b = Convert.ToInt32(Console.ReadLine());  
Console.Write("Bitte beliebes Zeichen auswählen: ");  
char c = Convert.ToChar(Console.ReadLine());  
for (int i = 0; i < b; i++) {  
    a = "";  
    if (i == 0 || i == b - 1) {  
        a = a.PadRight(b, c);  
        Console.WriteLine(a);  
    }  
    else {  
        a = a.PadRight(1, c) +  
            a.PadRight(b - 2) +  
            a.PadRight(1, c);  
        Console.WriteLine(a);  
    }  
}
```



```
# # # # # #  
#           #  
#           #  
#           #  
#           #  
# # # # # #
```

# Version 3 - Quadrat

- Lösung mit Console Write & Console Write Line

```
#####
#####
#####
###   ###
###   ###
###   ###
#####
#####
#####
```

```
int a = 7;    // Int32.Parse(Console.ReadLine());
char c = '#'; // Char.Parse(Console.ReadLine());

if (a < 7 || a > 21) {
    Console.WriteLine("Wähle eine Zahl zwischen 7 und 21");
    return;
}
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < a; j++) {
        Console.Write(c);
    }
    Console.WriteLine();
}
for (int i = 0; i < a - 6; i++) {
    for (int j = 0; j < 3; j++) {
        Console.Write(c);
    }
    for (int j = 0; j < a - 6; j++) {
        Console.Write(" ");
    }
    for (int j = 0; j < 3; j++) {
        Console.Write(c);
    }
    Console.WriteLine();
}
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < a; j++) {
        Console.Write(c);
    }
    Console.WriteLine();
}
```

# Version 3 - Quadrat

- Lösung mit StringBuilder

```
#####
#####
#####
###  ###
#####
#####
#####
```

```
#####
#####
#####
###  ###
###  ###
###  ###
#####
#####
#####
```

```
int a = 7; // Int32.Parse(Console.ReadLine());
char c = '#'; // Char.Parse(Console.ReadLine());
StringBuilder sb = new StringBuilder();
if (a < 7 || a > 21) {
    Console.WriteLine("Wähle eine Zahl zwischen 7 und 21");
    return;
}
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < a; j++) {
        sb = sb.Append(c);
    }
    sb = sb.Append("\n");
}
for (int i = 0; i < a - 6; i++) {
    for (int j = 0; j < 3; j++) {
        sb = sb.Append(c);
    }
    for (int j = 0; j < a - 6; j++) {
        sb = sb.Append(" ");
    }
    for (int j = 0; j < 3; j++) {
        sb = sb.Append(c);
    }
    sb = sb.Append("\n");
}
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < a; j++) {
        sb = sb.Append(c);
    }
    sb = sb.Append("\n");
}
Console.WriteLine(sb);
```



# Version 3 - Quadrat

- Lösung mit PadRight

```

string a;
int b = 0;
Console.Write("Bitte die Größe(Min: 7 Max: 22): ");
b = Convert.ToInt32(Console.ReadLine());
Console.Write("Bitte beliebes Zeichen auswählen: ");
char c = Convert.ToChar(Console.ReadLine());
for (int i = 0; i < b; i++) {
    a = "";
    if (i == 0 || i == 1 || i == 2 ||
        i == b - 1 || i == b - 2 || i == b - 3) {
        a = a.PadRight(b, c);
        Console.WriteLine(a);
    }
    else {
        a = a.PadRight(3, c)
            + a.PadRight(b - 6)
            + a.PadRight(3, c);
        Console.WriteLine(a);
    }
}

```

```

#####
#####
#####
###   ###
###   ###
###   ###
#####
#####
#####

```

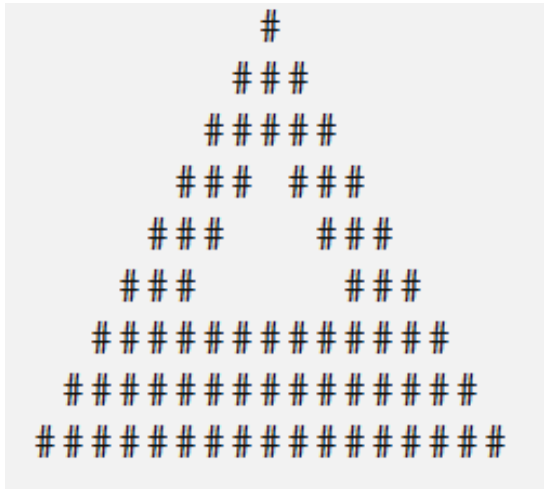
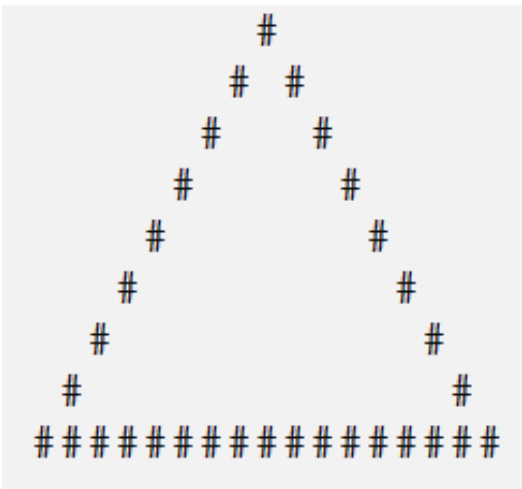
```

#####
#####
#####
###   ###
###   ###
###   ###
#####
#####
#####

```

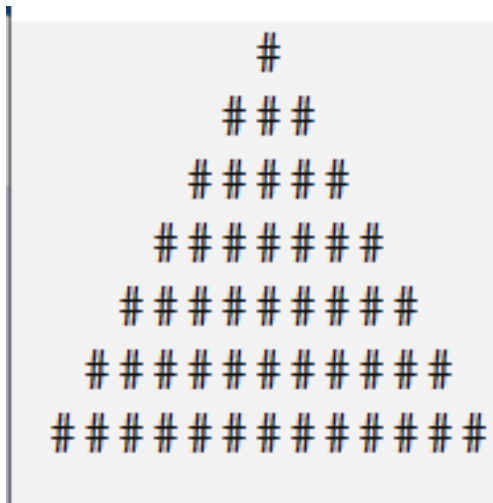
# Ausgabe der Dreiecke

- Version 1
- Version 2
- Version 3



# Version 1 - Dreieck

- Lösung mit Console Write
- Console Write Line



```
#
###
#####
#####
#####
#####
#####
```

```
int a = 7;      // Int32.Parse(Console.ReadLine());
char c = '#';   // Char.Parse(Console.ReadLine());
int b = 1;
//Für Zeile 1 .. 7
for (int i = 0; i < a; i++) {
    //gib 7/2 x ein Leerzeichen aus
    //(damit das erste Zeichen in der Mitte steht)
    for (int j = 0; j < (a - (b / 2)); j++) {
        Console.Write(' ');
    }
    //Schreibe zuerst 1 Zeichen in die Mitte, und

    for (int j = 0; j < b; j++) {
        Console.Write(c);
    }
    //erhöhe je Zeilendurchgang um 2 Zeichen
    b += 2;
    //Enter am Zeilenende
    Console.WriteLine();
}
```

# Version 1 - Dreieck

- Lösung mit  
StringBuilder



```
#
###
#####
#####
#####
```

```
int a = 5;          // Int32.Parse(Console.ReadLine());
char c = '#';       // Char.Parse(Console.ReadLine());
int b = 1;
StringBuilder sb = new StringBuilder();
for (int i = 0; i < a; i++) {
    for (int j = 0; j < (a - (b / 2)); j++) {
        sb = sb.Append(' ');
    }
    sb = sb.Append(c, b);
    b += 2;
    sb.AppendLine();
}
Console.WriteLine(sb);
```

# Version 1 - Dreieck

- Lösung mit new String

```
Bitte gebe eine Höhe zwischen 2 und 24 ein:
5
Bitte gebe ein Zeichen ein:
+
    ++
  ++++
++++++
+++++++
+++++++
+++++++
+++++++
```

```
//bool, ob die Höhe okay ist
bool okay = false;

//int für Höhe jetzt schon, weil wenn es in der Klammer wäre
//würde man es außerhalb nicht mehr verwenden können
int height;

//Höhe einlesen, solange bis der Wert passend ist
do{...}

Console.WriteLine("Bitte gebe ein Zeichen ein:");
//Zeichen wird eingelesen
char sign = Char.Parse(Console.ReadLine());

//counter, für die Zeichen, die jedes Mal zwei mehr werden (links und rechts)
int counter = 0;

//Schleife, die sich so oft wiederholt, wie das Dreieck groß ist (eingelesene Zahl)(Höhe - 1 weil
//sonst das Dreieck nicht ganz am Rand ist
//Jeder Durchgang macht eine Zeile des Dreiecks
for (int i = height - 1; i >= 0; i--)
{
    //Counter wird erhöht (nur um 1, da Zeichen unten zweimal ausgegeben werden
    counter++;

    //string für die Leerzeichen am Rand
    string edge = new string(' ', i);

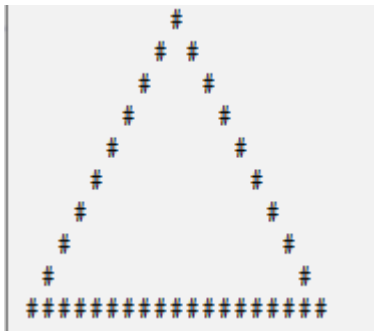
    //string für die Zeichen des Dreiecks
    string signs = new string(sign, counter);

    //string der die vorherigen Strings zusammenfügt (nicht nötig, da man es auch so
    //in cwl (Console.WriteLine) schreiben könnte, aber schöner): Leerzeichen außen + zweimal
    //die Zeichen (einmal für links, einmal für rechts) + wieder die Leerzeichen (Letzteres unnötig,
    //aber dann ist es symmetrisch
    string sum = edge + signs + signs + edge;

    //Ausgeben der Zeile des Dreiecks
    Console.WriteLine(sum);
}
```

# Version 2 - Dreieck

- Lösung mit Console Write
- Console Write Line

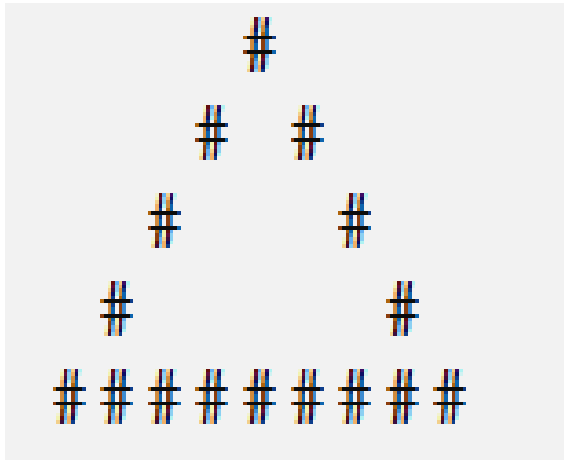


```
int a = 10;           //Int32.Parse(Console.ReadLine());
char c = '#';        // Char.Parse(Console.ReadLine());
int b = 1;

for (int i = 0; i < a; i++) {
    Console.Write(' ');
}
Console.Write(c);
b += 2;
Console.WriteLine();
for (int i = 0; i < a - 2; i++) {
    for (int j = 0; j < (a - (b / 2)); j++) {
        Console.Write(' ');
    }
    Console.Write(c);
    if (b - 2 > 0)
        for (int j = 0; j < b-2; j++) {
            Console.Write(' ');
        }
    Console.Write(c);
    b += 2;
    Console.WriteLine();
}
Console.Write(' ');
for (int j = 0; j < b; j++) {
    Console.Write(c);
}
Console.WriteLine();
```

# Version 2 - Dreieck

- Lösung mit StringBuilder



```
int a = 5;           //Int32.Parse(Console.ReadLine());
char c = '#';        // Char.Parse(Console.ReadLine());
int b = 1;
StringBuilder sb = new StringBuilder();
sb = sb.Append(' ', a);
sb = sb.Append(c);
b += 2;
sb.Append("\n");
for (int i = 0; i < a - 2; i++) {
    for (int j = 0; j < (a - (b / 2)); j++) {
        sb = sb.Append(' ');
    }
    sb = sb.Append(c);
    if (b - 2 > 0)
        sb = sb.Append(' ', b - 2);
    sb = sb.Append(c);
    b += 2;
    sb.Append("\n");
}
sb = sb.Append(' ');
sb = sb.Append(c, b);
Console.WriteLine(sb);
```





# Version 3 - Dreieck

- Lösung mit
- Console Write
- Console Write Line



```

int a = 9;        // Int32.Parse(Console.ReadLine());
char c = '#';     // Char.Parse(Console.ReadLine());
int b = 1;
StringBuilder sb = new StringBuilder();
if (a < 7 || a > 21) {
    Console.WriteLine("Wähle eine Zahl zwischen 7 und 21");
    return;
}
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < a-(b/2); j++) {
        Console.Write(' ');
    }
    for (int j = 0; j < b; j++) {
        Console.Write(c);
    }
    b += 2;
    Console.WriteLine();
}
for (int i = 0; i < a - 6; i++) {
    for (int j = 0; j < (a - (b / 2)); j++) {
        Console.Write(' ');
    }
    for (int j = 0; j < 3; j++) {
        Console.Write(c);
    }
    if (b - 2 > 0)
        for (int j = 0; j < b-6; j++) {
            Console.Write(' ');
        }
    for (int j = 0; j < 3; j++) {
        Console.Write(c);
    }
    b += 2;
    Console.WriteLine();
};
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < a - (b / 2); j++) {
        Console.Write(' ');
    }
    for (int j = 0; j < b; j++) {
        Console.Write(c);
    }
    b += 2;
    Console.WriteLine();
}

```

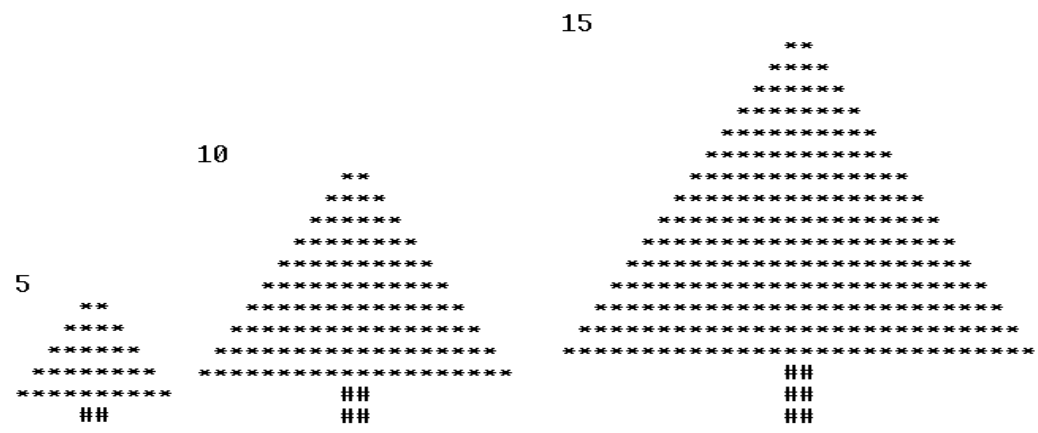
# Version 3 - Dreieck

- Lösung mit StringBuilder



```
int a = 9;        // Int32.Parse(Console.ReadLine());
char c = '#';     // Char.Parse(Console.ReadLine());
int b = 1;
StringBuilder sb = new StringBuilder();
if (a < 7 || a > 21) {
    Console.WriteLine("Wähle eine Zahl zwischen 7 und 21");
    return;
}
for (int i = 0; i < 3; i++) {
    sb = sb.Append(' ', a - (b / 2));
    sb = sb.Append(c, b);
    b += 2;
    sb.Append("\n");
}
for (int i = 0; i < a - 6; i++) {
    for (int j = 0; j < (a - (b / 2)); j++) {
        sb = sb.Append(' ');
    }
    sb = sb.Append(c, 3);
    if (b - 2 > 0)
        sb = sb.Append(' ', b - 6);
    sb = sb.Append(c, 3);
    b += 2;
    sb.Append("\n");
};
for (int i = 0; i < 3; i++) {
    sb = sb.Append(' ', a - (b / 2));
    sb = sb.Append(c, b);
    sb = sb.Append('\n');
    b += 2;
}
Console.WriteLine(sb);
```





Version 1 : einfach

# Einlesen der Höhe

```
while (height < 3)
{
    Console.WriteLine("Bitte geben Sie eine größere Zahl ein!");
    height = Convert.ToInt16(Console.ReadLine());
}
```

# Drucken der Krone

```
// Krone
int spaces = height - 1;
for (int i = 1; i <= height; i++)
{
    for (int j = 0; j < spaces; j++)
    {
        Console.Write(" ");
    }
    for (int k = 0; k < (2 * i); k++)
    {
        Console.Write("*");
    }
    Console.WriteLine("");
    spaces--;
}
```

# Drucken des Stamms

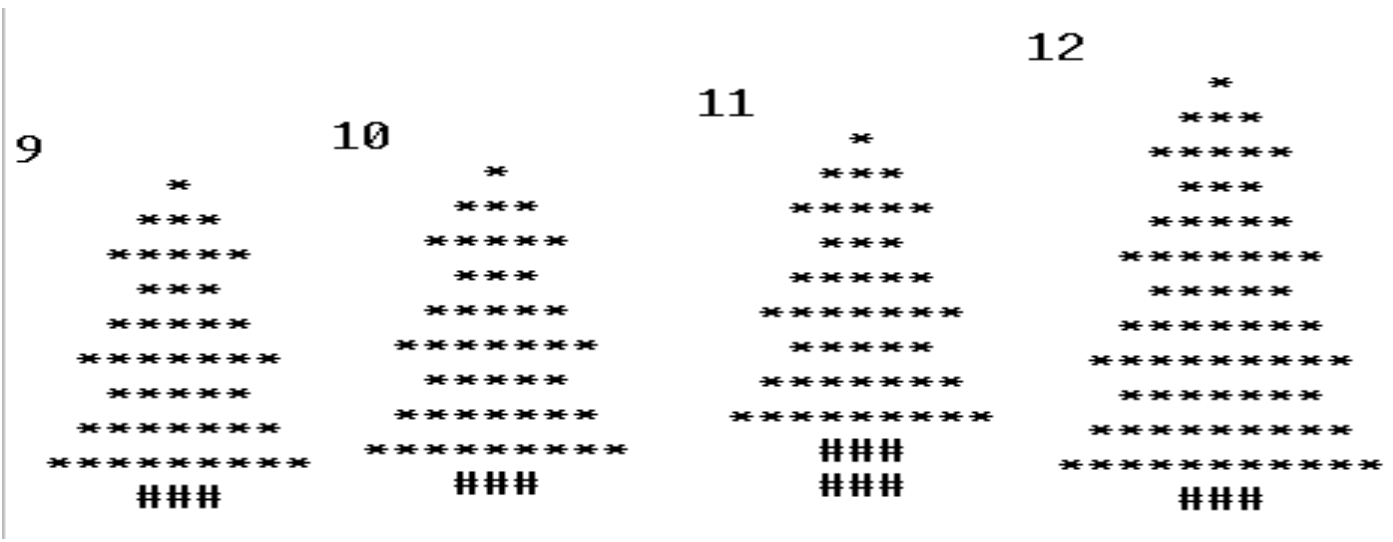
```
// Stamm
int trunkHeight = (height/5==0) ? 1 : height/5;
for (int i = 0; i < trunkHeight; i++)
{
    for (int j = 0; j < (height - 1); j++)
    {
        Console.Write(" ");
    }
    Console.WriteLine("##");
}
```

# Fragezeichen Doppelpunkt Operator

- ```
if (Bedingung) {  
    Variable = "Bedingung erfüllt";  
} else {  
    Variable = "Bedingung nicht erfüllt";  
}
```
- Diesen Code kann man auch kürzer schreiben, in dem man ein Fragezeichen (?) und einen Doppelpunkt (:) benutzt:

```
Variable = (Bedingung) ? "Bedingung erfüllt" :  
"Bedingung nicht erfüllt";
```





Version 2 : gestuft

```
Console.WriteLine("Bitte geben Sie die gewünschte Höhe des Baumes ein\n");
int lines = Int32.Parse(Console.ReadLine());
String input = "";
int length = input.Length;
int rowlength = 0;
int padding = 0;
//Maximale Anzahl der Sterne/Reihe im Baum
int max_chars = 3 + 2 * lines / 3;
int count = 1;
for (int i = 0; i < lines / 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        //Eine Sternzeile erstellen
        input = string.Empty.PadRight(count, '*');
        rowlength = max_chars;
        length = input.Length;
        padding = (rowlength - length) / 2 + 1;
        string sternzeile = input.PadLeft(padding + length, ' ');
        Console.WriteLine(sternzeile);
        count += 2;
    }
    count -= 4;
}
```

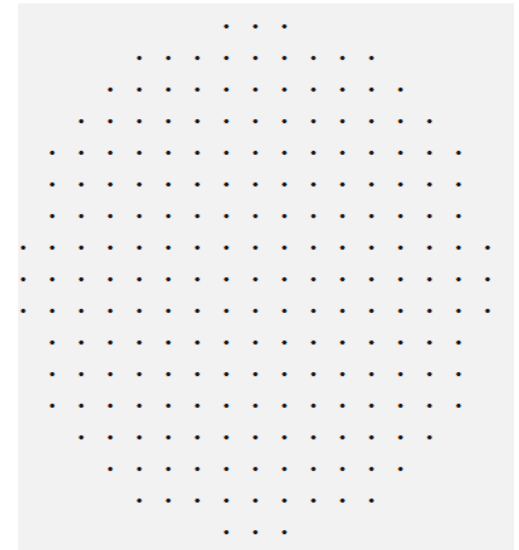
## Version 2

# Stammhöhe ausgeben

```
//Baumstamm hat mindestens eine Höhe von 1
int stammhoehe = lines % 3 == 0 ? 1 : lines % 3;

for (int i = 0; i < stammhoehe; i++)
{
    rowlength = max_chars;
    input = "###";
    length = input.Length;
    padding = (rowlength - length) / 2 + 1;

    Console.WriteLine(input.PadLeft(padding + length, ' '));
}
Console.WriteLine();
```



# Kreis zeichnen

Einlesen einer Zahl und Ausgabe eines Kreises

# Kreis zeichnen

```

      x x x
    x x x x x
  x x x x x x x
x x x x x x x x x
x x x x x x x x x
x x x x x x x x x
  x x x x x x x
    x x x x x
      x x x
  
```

```

static void drawCircle() {
    int r = 4;
    char c = 'x';
    // Consider a rectangle of size N*N
    int N = 2 * r + 1;
    int x, y; // Coordinates inside the rectangle
    // Draw a square of size N*N.
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            // Start from the left most corner point
            x = i - r;
            y = j - r;
            // If this point is inside the circle, print it
            if (x * x + y * y <= r * r + 1)
                Console.Write(c);
            else
                // If outside the circle, print space
                Console.Write(" ");

            Console.Write(" ");
        }
        Console.WriteLine();
    }
}
  
```