|  |
| --- |
| SEW |
| Weihnachtsformen |
| Gehirnschmalz entwickeln, Files & StringBuilder verinnerlichen |

|  |
| --- |
| Felix  29.12.2021  [4h] |

# Inhaltsverzeichnis

Inhalt

[Inhaltsverzeichnis 1](#_Toc56699300)

[Angabe 1 1](#_Toc56699301)

[Quellcode 1 1](#_Toc56699302)

[Testfälle 1 1](#_Toc56699303)

[Angabe 2 1](#_Toc56699304)

[Klassendiagramm 2 1](#_Toc56699305)

[Quellcode 2 1](#_Toc56699306)

[Testfälle 2 1](#_Toc56699307)

[Angabe 3 1](#_Toc56699308)

[Klassendiagramm 3 1](#_Toc56699309)

[Quellcode 3 1](#_Toc56699310)

[Testfälle 3 1](#_Toc56699311)

Rechtsklick und Inhaltsverzeichnis aktualisieren…

Lösche die Teilbereiche je Aufgabenstellung die nicht benötigt werden, gibt es nur ein Beispiel reicht es einmal die Überschriften zu nutzen.

Öffne die Formatvorlage (nicht neues Dokument – sondern öffnen anklicken – lösche alle Erklärungen die du nicht benötigst – und setze unter Entwurf ein eigenes Design dass dir gut gefällt. Nutze weiterhin deine eigene Vorlage, falls du schon eine hast.)

# Angabe 1 <<…>>

Erstelle unterschiedliche Formen – schreibe jede Form in eine Datei

Öffne den Foliensatz Console\_String\_StringBuilder\_Formen

Gehe zu Folie 52 Formen darstellen - Schreibe diese Formen in eine Datei

Löse die Aufgaben

* Quadrat ausgeben

Schreibe die Formen in eine Textdatei –   
probiere die Aufgaben ohne Musterlösung im Foliensatz!

# Theorie 1 <<…>>

Unterhalb sehen Sie die 0815-Standardzeichenvariante von einer beliebigen Form (wahrscheinlich nicht alle…).

Fügen Sie einfach bei dem rot kommentierten Satz eine Mathematische Funktion zur Berechnung von b ein und die Form sollte richtig ausgegeben werden.

Außerdem setzen Sie die gleiche Formel bei max ein und statt i bei b setzen Sie die Zeile ein, in der der Maximalwert ist… (Bei Kreis mehr dazu…)

static string Circle(int size)

{

StringBuilder sb = new StringBuilder();

if(size < 10) size = 10;

double max = 0;

// heftige Berechnung des Maximalwertes (Zeile mit meisten Zeichen).

for (int i = 0; i < size + 1; i++)

{

// heftige Mathematische Formel zur berechnung von b

for (int j = 0; j < (max - b) / 2; j++) // Leerzeichen

{

sb.Append(" ");

}

for (int j = 0; j < b; j++) // eigentlicher Kreis

{

sb.Append("#");

}

sb.AppendLine();

}

return sb.ToString();

# }

# Quellcode 1 <<…>>

using System.Text;

static void Write(string path, string text)

{

using(StreamWriter sw = new StreamWriter(path, true))

{

sw.WriteLine();

sw.WriteLine(text);

}

}

static void DeleteOldFile(string path)

{

using(StreamWriter sw = new StreamWriter(path))

{

sw.Write("");

}

}

static string Rectangle(int x, int y)

{

StringBuilder sb = new StringBuilder();

for (int i = 0; i < y; i++) // Rechteck

{

for (int j = 0; j < x; j++)

{

sb.Append("\*");

}

if(i != y - 1)

sb.AppendLine();

}

return sb.ToString();

}

static string Square(int size)

{

return Rectangle(size, size); // Quadrat hat gleiche Breite wie Länge

}

static string RectangleEmpty(int x, int y, int edge)

{

StringBuilder sb = new StringBuilder();

for (int i = 0; i < edge; i++) // oberer Rand

{

for (int j = 0; j < x; j++)

{

sb.Append("\*");

}

if (i != y - 1)

sb.AppendLine();

}

for (int i = 0; i < y - 2 \* edge; i++) // mittlerer Teil (zuerst Rand, dann leer, dann Rand)

{

for (int j = 0; j < x; j++)

{

if (j < edge || j >= x - edge)

sb.Append("\*");

else sb.Append(" ");

}

if (i != y - 1)

sb.AppendLine();

}

for (int i = 0; i < edge; i++) // unterer Rand

{

for (int j = 0; j < x; j++)

{

sb.Append("\*");

}

if (i != edge - 1)

sb.AppendLine();

}

return sb.ToString();

}

static string SquareEmpty(int size, int edge)

{

return RectangleEmpty(size, size, edge); // Quadrat hat gleiche Breite wie Länge

}

DeleteOldFile("rectangles.txt");

Write("rectangles.txt", Rectangle(3, 4));

Write("rectangles.txt", "----------------------------------");

Write("rectangles.txt", Square(5));

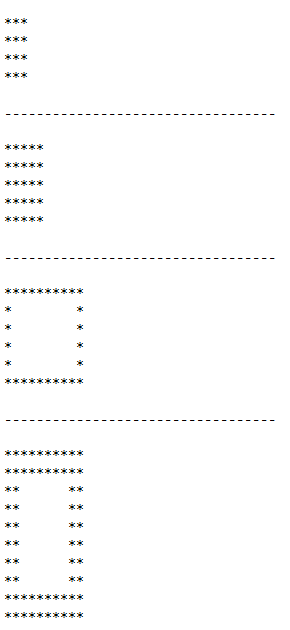
Write("rectangles.txt", "----------------------------------");

Write("rectangles.txt", RectangleEmpty(10, 6, 1));

Write("rectangles.txt", "----------------------------------");

# Write("rectangles.txt", SquareEmpty(10, 2));

# Testfälle 1 <<…>>



# Angabe 2 <<…>>

Erstelle unterschiedliche Formen – schreibe jede Form in eine Datei

Öffne den Foliensatz Console\_String\_StringBuilder\_Formen

Gehe zu Folie 52 Formen darstellen - Schreibe diese Formen in eine Datei

Löse die Aufgaben

* Dreieck ausgeben

Schreibe die Formen in eine Textdatei –   
probiere die Aufgaben ohne Musterlösung im Foliensatz!

# Quellcode 2 <<…>>

using System.Text;

static void Write(string path, string text)

{

using (StreamWriter sw = new StreamWriter(path, true))

{

sw.WriteLine();

sw.WriteLine(text);

}

}

static void DeleteOldFile(string path)

{

using (StreamWriter sw = new StreamWriter(path))

{

sw.Write("");

}

}

static string Triangle(int size)

{

StringBuilder sb = new StringBuilder();

if (size % 2 == 0) // size muss ungerade sein

size++;

int b = 1; // b = Anzahl an Zeichen (\*) in einer Zeile

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size - (b / 2); j++)

{

sb.Append(" ");

}

for (int j = 0; j < b; j++)

{

sb.Append("\*");

}

b += 2;

sb.AppendLine();

}

return sb.ToString();

}

static string TriangleEmpty(int size, int edge)

{

StringBuilder sb = new StringBuilder();

if (size % 2 == 0)

size++;

int b = 1;

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size - (b / 2); j++)

{

sb.Append(" ");

}

for (int j = 0; j < edge - 1; j++)

{

sb.Append(" ");

}

for (int j = 0; j < b; j++)

{

if (j < edge || j >= b - edge)

sb.Append("\*");

else sb.Append(" ");

}

b += 2;

sb.AppendLine();

}

for (int i = 0; i < edge; i++)

{

for (int j = 0; j < edge - i - 1; j++)

{

sb.Append(" ");

}

for (int j = 0; j < b; j++)

{

sb.Append("\*");

}

b += 2;

sb.AppendLine();

}

return sb.ToString();

}

DeleteOldFile("triangles.txt");

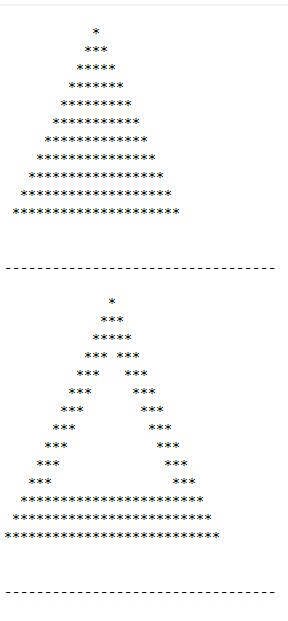
Write("triangles.txt", Triangle(11));

Write("triangles.txt", "----------------------------------");

Write("triangles.txt", TriangleEmpty(11, 3));

Write("triangles.txt", "----------------------------------");

# Testfälle 2 <<…>>



# Angabe 3 <<…>>

Erstelle unterschiedliche Formen – schreibe jede Form in eine Datei

Öffne den Foliensatz Console\_String\_StringBuilder\_Formen

Gehe zu Folie 52 Formen darstellen - Schreibe diese Formen in eine Datei

Löse die Aufgaben

* Einfacher Tannenbaum
* Gestufter Tannenbaum

Schreibe die Formen in eine Textdatei –   
probiere die Aufgaben ohne Musterlösung im Foliensatz!

# Quellcode 3 <<…>>

using System.Text;

static void Write(string path, string text)

{

using (StreamWriter sw = new StreamWriter(path, true))

{

sw.WriteLine();

sw.WriteLine(text);

}

}

static void DeleteOldFile(string path)

{

using (StreamWriter sw = new StreamWriter(path))

{

sw.Write("");

}

}

static string Triangle(int von, int bis, int max)

{

StringBuilder sb = new StringBuilder();

// von & bis müssen entweder beide gerade oder ungerade sein, weil b immer um 2 erhöht wird

if (von % 2 == 0) // Fall 1: von & bis sind gerade

{

if (bis % 2 != 0) bis++;

}

else if (bis % 2 == 0) bis++; // Fall 2: von & bis sind ungerade

int size = (bis - von) / 2 + 1;

int b = von; // b = Anzahl an Zeichen (\*) in einer Zeile

for (int i = 0; i < size; i++)

{

for (int j = 0; j < (max - bis) / 2; j++) // Damit oberen Dreiecke mittig in Bezug auf unterstem Dreieck sind.

// Ansonsten wäre die letzte Zeile des ersten Dreiecks des Baumes ganz am linken Rand...

{

sb.Append(" ");

}

for (int j = 0; j < (bis - b) / 2; j++) // Leerzeichen links von Baum

{

sb.Append(" ");

}

for (int j = 0; j < b; j++) // eigentlicher Baum

{

sb.Append("\*");

}

b += 2; // 2, weil links und rechts, ansonsten nicht symmetrisch

sb.AppendLine();

}

return sb.ToString();

}

static string Tree(int size, int x) //x = Anzahl\_wie\_viele\_b\_plus\_pro\_Fluegel

{

StringBuilder sb = new StringBuilder();

while (size % x != 0) size++;

int b = 1;

int max = 1 + (size / x - 1) \* 2 + (x-1) \* 2;

for (int i = 0; i < size / x; i++)

{

sb.Append(Triangle(b, b + (x-1) \* 2, max));

b += 2;

}

for (int i = 0; i < size / 6; i++)

{

for (int j = 0; j < max/3; j++)

{

sb.Append(" ");

}

int dicke = max / 3;

if (max % 3 != 0) dicke++;

for (int j = 0; j < dicke; j++)

{

sb.Append("#");

}

sb.AppendLine();

}

return sb.ToString();

}

DeleteOldFile("trees.txt");

Write("trees.txt", Tree(38, 7));

Write("trees.txt", "--------------------------------------------------------------");

Write("trees.txt", Tree(16, 5));

Write("trees.txt", "--------------------------------------------------------------");

Write("trees.txt", Tree(14, 8));

Write("trees.txt", "--------------------------------------------------------------");

Write("trees.txt", Tree(69, 3));

Write("trees.txt", "--------------------------------------------------------------");

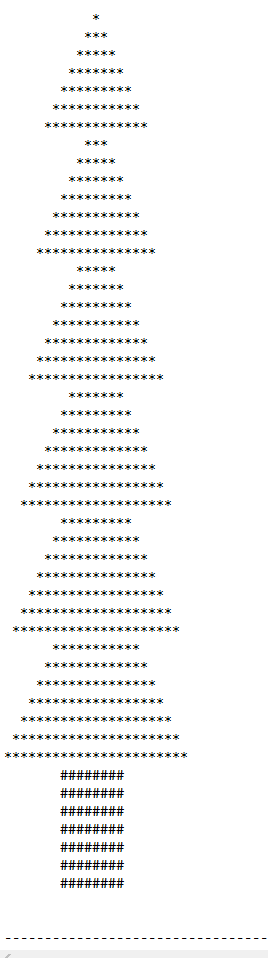
Write("trees.txt", Tree(20, 2));

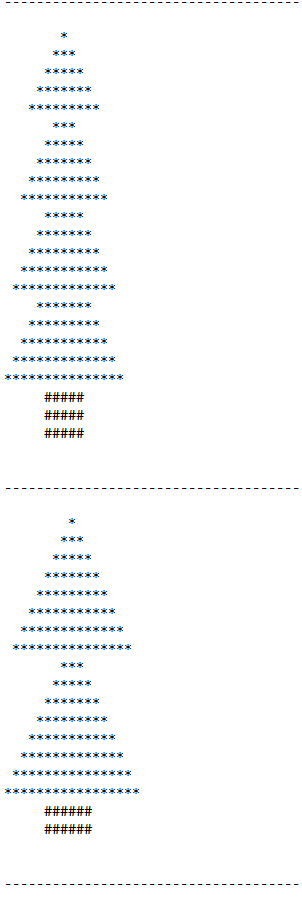
Write("trees.txt", "--------------------------------------------------------------");

Write("trees.txt", Tree(11, 1));

Write("trees.txt", "--------------------------------------------------------------");

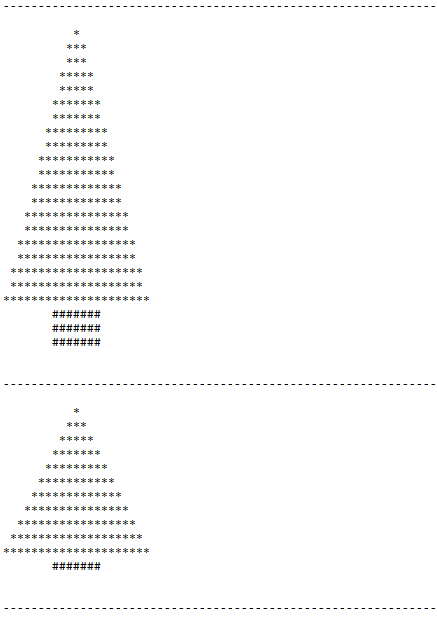
# Testfälle 3 <<…>>





Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



# Angabe 4 <<…>>

Erstelle unterschiedliche Formen – schreibe jede Form in eine Datei

Öffne den Foliensatz Console\_String\_StringBuilder\_Formen

Gehe zu Folie 52 Formen darstellen - Schreibe diese Formen in eine Datei

Löse die Aufgaben

* Kreis/e ausgeben -> Schneemann ausgeben

Schreibe die Formen in eine Textdatei –   
probiere die Aufgaben ohne Musterlösung im Foliensatz!

# Quellcode 4 <<…>>

using System.Text;

static void Write(string path, string text)

{

using (StreamWriter sw = new StreamWriter(path, true))

{

sw.WriteLine();

sw.WriteLine(text);

}

}

static void DeleteOldFile(string path)

{

using (StreamWriter sw = new StreamWriter(path))

{

sw.Write("");

}

}

static string Circle(int size)

{

StringBuilder sb = new StringBuilder();

if(size < 10) size = 10;

double max = Math.Round(Math.Sqrt(Math.Pow(size / 2, 2) - Math.Pow(size / 2 - size / 2, 2)) \* 4.4);

for (int i = 0; i < size + 1; i++)

{

double b = Math.Round(Math.Sqrt(Math.Pow(size / 2, 2) - Math.Pow(i - size / 2, 2)) \* 4.4);

for (int j = 0; j < (max - b) / 2; j++) // Leerzeichen

{

sb.Append(" ");

}

for (int j = 0; j < b; j++) // eigentlicher Kreis

{

sb.Append("#");

}

sb.AppendLine();

}

return sb.ToString();

}

static string CircleMax(int size, double biggestsize)

{

StringBuilder sb = new StringBuilder();

if (size < 10) size = 10;

double max = Math.Round(Math.Sqrt(Math.Pow(size / 2, 2) - Math.Pow(size / 2 - size / 2, 2)) \* 4.4);

for (int i = 0; i < size + 1; i++)

{

double b = Math.Round(Math.Sqrt(Math.Pow(size / 2, 2) - Math.Pow(i - size / 2, 2)) \* 4.4);

for (int j = 0; j < biggestsize - size; j++) // Damit oberster und mittlerer Kreis zentriert

{

sb.Append(" ");

}

for (int j = 0; j < (max - b) / 2; j++) // Leerzeichen

{

sb.Append(" ");

}

for (int j = 0; j < b; j++) // eigentlicher Kreis

{

sb.Append("#");

}

if(i < size - 2) // Damit Schneemannkugeln zusammenkleben

sb.AppendLine();

}

return sb.ToString();

}

static string Snowman(int size)

{

StringBuilder sb = new StringBuilder();

double max = Math.Round(Math.Sqrt(Math.Pow(size / 3 \* 1.4 / 2, 2) - Math.Pow(size / 3 \* 1.4 / 2 - size / 3 \* 1.4 / 2, 2)) \* 4.4) - 5;

// Hut

for (int i = 0; i < size / 6; i++)

{

for (int j = 0; j < (max - (size / 3 \* 0.5)) / 2; j++) // Damit Hut zentriert

{

sb.Append(" ");

}

for (int j = 0; j < size / 3 \* 0.5; j++)

{

sb.Append("#");

}

sb.AppendLine();

}

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < (max - (size / 3 \* 1.2)) / 2; j++) // Damit Hut zentriert

{

sb.Append(" ");

}

for (int j = 0; j < size / 3 \* 1.2; j++)

{

sb.Append("#");

}

if(i < 1)

sb.AppendLine();

}

max = size / 3 \* 1.4;

// eigentlicher Schneeman

for (double i = 0.6; i <= 1.4 ; i+=.4)

{

sb.Append(CircleMax((int)Math.Round(size / 3 \* i), max));

}

return sb.ToString();

}

DeleteOldFile("circles.txt");

DeleteOldFile("snowmans.txt");

Write("circles.txt", Circle(40));

Write("circles.txt", "--------------------------------------------------------------");

Write("circles.txt", Circle(100));

Write("circles.txt", "--------------------------------------------------------------");

Write("circles.txt", Circle(15));

Write("circles.txt", "--------------------------------------------------------------");

Write("circles.txt", Circle(5));

Write("circles.txt", "--------------------------------------------------------------");

Write("circles.txt", Circle(1));

Write("circles.txt", "--------------------------------------------------------------");

Write("circles.txt", Circle(77));

Write("circles.txt", "--------------------------------------------------------------");

Write("snowmans.txt", Snowman(20));

Write("snowmans.txt", "--------------------------------------------------------------");

Write("snowmans.txt", Snowman(50));

Write("snowmans.txt", "--------------------------------------------------------------");

Write("snowmans.txt", Snowman(77));

Write("snowmans.txt", "--------------------------------------------------------------");

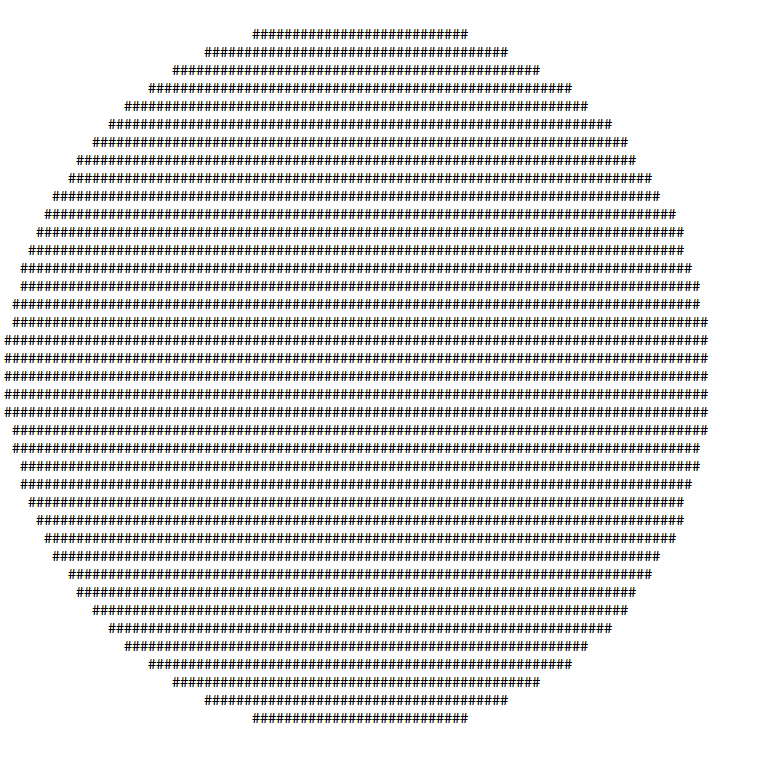
Write("snowmans.txt", Snowman(100));

Write("snowmans.txt", "--------------------------------------------------------------");

Write("snowmans.txt", Snowman(200));

Write("snowmans.txt", "--------------------------------------------------------------");

# Testfälle 4 <<…>>

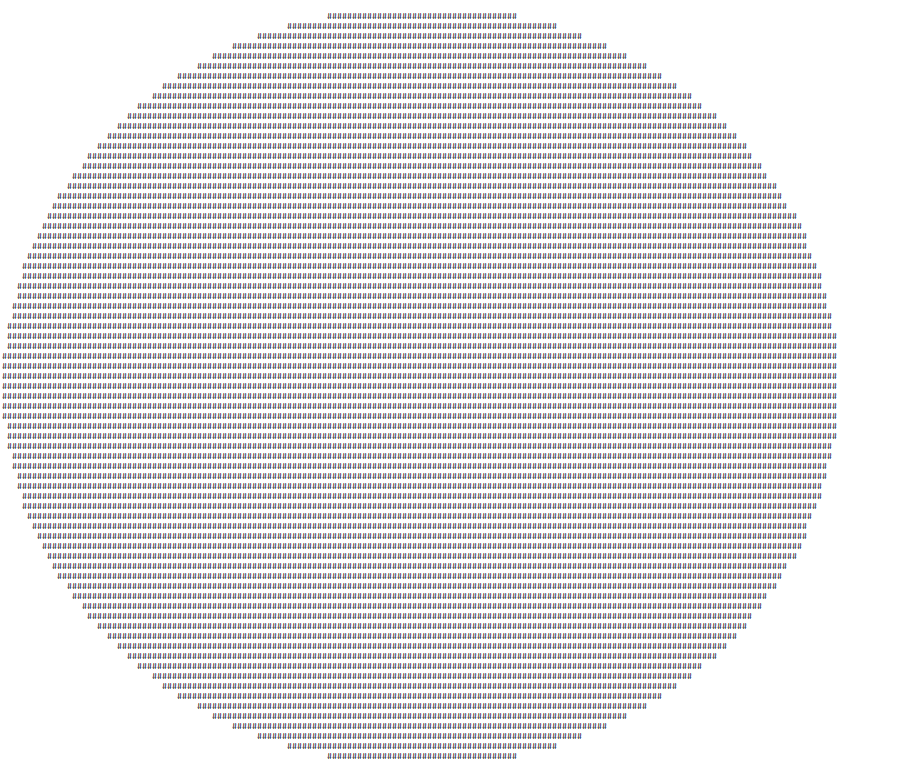


Ein Bild, das Kuppel enthält.

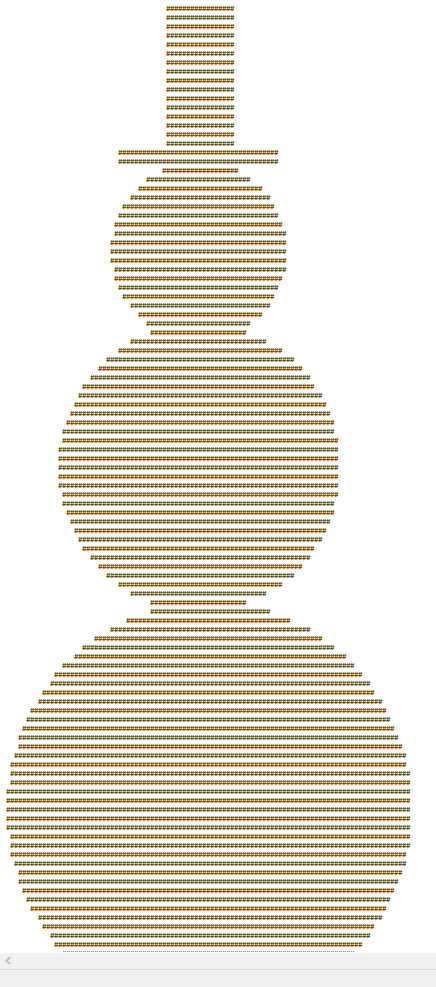
Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung







Kleine Schneemänner schauen nicht so gut aus, weil sie so schnell schmelzen, dass man sie nicht fotografieren kann…

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

# Angabe 5 <<…>>

Erstelle unterschiedliche Formen – schreibe jede Form in eine Datei

Öffne den Foliensatz Console\_String\_StringBuilder\_Formen

Gehe zu Folie 52 Formen darstellen - Schreibe diese Formen in eine Datei

Löse die Aufgaben

* Stern ausgeben

Schreibe die Formen in eine Textdatei –   
probiere die Aufgaben ohne Musterlösung im Foliensatz!

# Quellcode 5 <<…>>

using System.Text;

static void Write(string path, string text)

{

using (StreamWriter sw = new StreamWriter(path, true))

{

sw.WriteLine();

sw.WriteLine(text);

}

}

static void DeleteOldFile(string path)

{

using (StreamWriter sw = new StreamWriter(path))

{

sw.Write("");

}

}

static string Star(int size)

{

StringBuilder sb = new StringBuilder();

if (size < 10) size = 10;

double max = -8 \* size / 2 + 100;

for (int i = 0; i < size; i++)

{

double b = 0;

if (i >= 0 && i < size \* 2 / 5)

b = i / 2;

else if (i >= size \* 2 / 5 && i < size / 2)

b = 8 \* i - 60;

else if (i >= size / 2 && i < size \* 3 / 5)

b = -8 \* i + 100;

else if (i >= size \* 3 / 5 && i <= size)

b = -i / 2 + 10;

Console.WriteLine(b);

for (int j = 0; j < (max - b) / 2; j++) // Leerzeichen

{

sb.Append(" ");

}

for (int j = 0; j < b; j++) // eigentlicher Stern

{

sb.Append("#");

}

sb.AppendLine();

}

return sb.ToString();

}

DeleteOldFile("stars.txt");

Write("stars.txt", Star(40));

Write("stars.txt", "--------------------------------------------------------------");

Write("stars.txt", Star(100));

Write("stars.txt", "--------------------------------------------------------------");

Write("stars.txt", Star(15));

Write("stars.txt", "--------------------------------------------------------------");

Write("stars.txt", Star(5));

Write("stars.txt", "--------------------------------------------------------------");

Write("stars.txt", Star(1));

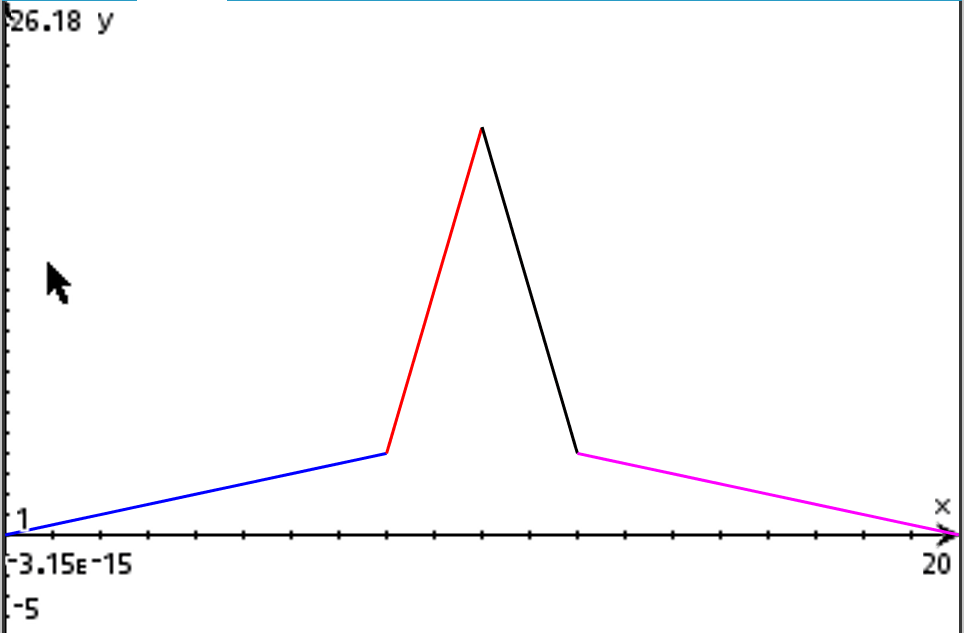
Write("stars.txt", "--------------------------------------------------------------");

Write("stars.txt", Star(77));

Write("stars.txt", "--------------------------------------------------------------");

# Testfälle 5 <<…>>

Mathematisch betrachtet muss b immer auf den y-Wert an der Stelle i gesetzt werden. Diese Funktion beschreibt dann den Stern:



Praktisch gesehen, passiert nicht das, was ich will das passieren sollte (ich suche später meinen Fehler (das ist nur eine Zwischenabgabe)):

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

# Angabe 6 <<…>>

Erstelle unterschiedliche Formen – schreibe jede Form in eine Datei

Öffne den Foliensatz Console\_String\_StringBuilder\_Formen

Gehe zu Folie 52 Formen darstellen - Schreibe diese Formen in eine Datei

Löse die Aufgaben

* Form deiner Wahl – Angabe & Lösung erstellen

Schreibe die Formen in eine Textdatei –   
probiere die Aufgaben ohne Musterlösung im Foliensatz!

VASE!

# Quellcode 6 <<…>>

using System.Text;

static void Write(string path, string text)

{

using (StreamWriter sw = new StreamWriter(path, true))

{

sw.WriteLine();

sw.WriteLine(text);

}

}

static void DeleteOldFile(string path)

{

using (StreamWriter sw = new StreamWriter(path))

{

sw.Write("");

}

}

static string Vase(int size)

{

StringBuilder sb = new StringBuilder();

if (size < 10) size = 10;

double max = 0;

for (int i = 0; i < size; i++)

{

double b = Math.Abs(Math.Round(-9.31 / (Math.Pow(size, 2)) \* Math.Pow(i, 3) + 12.883 / size \* Math.Pow(i, 2) - 3.723 \* i + 0.5 \* size));

max = Math.Max(max, b);

}

for (int i = 0; i < size; i++)

{

double b = Math.Abs(Math.Round(-9.31 / (Math.Pow(size, 2)) \* Math.Pow(i, 3) + 12.883 / size \* Math.Pow(i, 2) - 3.723 \* i + 0.5 \* size));

Console.WriteLine(b);

for (int j = 0; j < (max - b) / 2; j++) // Leerzeichen

{

sb.Append(" ");

}

for (int j = 0; j < b; j++) // eigentlicher Stern

{

sb.Append("=");

}

sb.AppendLine();

}

return sb.ToString();

}

DeleteOldFile("vases.txt");

Write("vases.txt", Vase(40));

Write("vases.txt", "--------------------------------------------------------------");

Write("vases.txt", Vase(100));

Write("vases.txt", "--------------------------------------------------------------");

Write("vases.txt", Vase(15));

Write("vases.txt", "--------------------------------------------------------------");

Write("vases.txt", Vase(5));

Write("vases.txt", "--------------------------------------------------------------");

Write("vases.txt", Vase(1));

Write("vases.txt", "--------------------------------------------------------------");

Write("vases.txt", Vase(77));

Write("vases.txt", "--------------------------------------------------------------");

# Testfälle 6 <<…>>

Man kann jede Form mit dieser Method bilden, solange bei der Form in einer Zeile nur eine zusammenhängende Gruppe an Zeichen in der Mitte der Zeile ist.





