# Programmierdokumentation Caesar Verschlüsselung

Roasrio-Francesco Polito
Tobias Dolfen

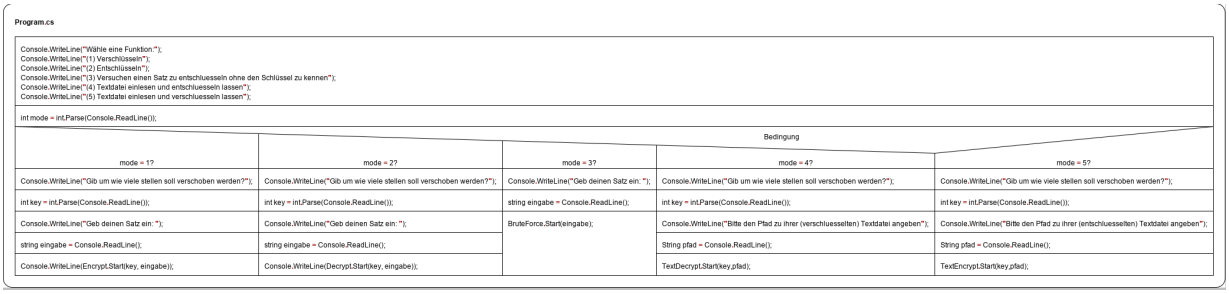
4. Februar 2019

# Inhaltsverzeichnis

# 1 Program.cs

## 1.1 main

Die Main Methode stellt den Einstieg in das Programm dar. Zu Beginn wird der Benutzer gefragt welchen Modus er nutzen möchte. Der gewählte Modus wird in einer integer Variable gespeichert worauf hin der gewünschte Modus gestartet wird.



Program.cs

```
Console.WriteLine("Wähle eine Funktion:");
Console.WriteLine("(1) Verschlüsseln");
Console.WriteLine("(2) Entschlüsseln");
Console.WriteLine("(3) Versuchen einen Satz zu entschluesseln ohne den Schlüssel zu kennen");
Console.WriteLine("(4) Textdatei einlesen und entschluesseln lassen");
Console.WriteLine("(5) Textdatei einlesen und verschluesseln lassen");

int mode = int.Parse(Console.ReadLine());
```

| Bedingung | | | | |
|---|---|---|---|---|
| mode = 1? | mode = 2? | mode = 3? | mode = 4? | mode = 5? |
| Console.WriteLine("Gib um wie viele stellen soll verschoben werden?"); | Console.WriteLine("Gib um wie viele stellen soll verschoben werden?"); | Console.WriteLine("Geb deinen Satz ein: "); | Console.WriteLine("Gib um wie viele stellen soll verschoben werden?"); | Console.WriteLine("Gib um wie viele stellen soll verschoben werden?"); |
| int key = int.Parse(Console.ReadLine()); | int key = int.Parse(Console.ReadLine()); | string eingabe = Console.ReadLine(); | int key = int.Parse(Console.ReadLine()); | int key = int.Parse(Console.ReadLine()); |
| Console.WriteLine("Geb deinen Satz ein: "); | Console.WriteLine("Geb deinen Satz ein: "); | BruteForce.Start(eingabe); | Console.WriteLine("Bitte den Pfad zu ihrer (verschluesselten) Textdatei angeben"); | Console.WriteLine("Bitte den Pfad zu ihrer (entschluesselten) Textdatei angeben"); |
| string eingabe = Console.ReadLine(); | string eingabe = Console.ReadLine(); | | String pfad = Console.ReadLine(); | String pfad = Console.ReadLine(); |
| Console.WriteLine(Encrypt.Start(key, eingabe)); | Console.WriteLine(Decrypt.Start(key, eingabe)); | | TextDecrypt.Start(key,pfad); | TextEncrypt.Start(key,pfad); |

Abbildung 1.1: Program.CS

## 1.2 Verschlüsseln

Wenn der „Verschlüsseln Modus“ gewählt wird, wird der Benutzer als Erstes gefragt, um wie viele Stellen er die Nachricht verschieben will. Die eingegebene Zahl wird in einem integer „key“ gespeichert. Im Anschluss daran wird der Benutzter nach dem gewünschten Satz gefragt, welcher verschlüsselt werden soll. Der eingegebene Satz wird in einem string „eingabe“ gespeichert. Nachdem der Benutzer den Satz eingegeben hat wird die Methode „Encrypt.Start“[1] aufgerufen. Diese bekommt die Werte „key“ und „eingabe“ übergeben.

## 1.3 Entschlüsseln

Wenn der „Entschlüsseln Modus“ gewählt wird, wird der Benutzer als erstes gefragt, um wie viele Stellen er die Nachricht verschieben will. Die eingegebene Zahl wird in einem integer „key“ gespeichert. Im Anschluss daran wird der Benutzter nach dem gewünschten Satz gefragt, welcher verschlüsselt werden soll. Der eingegebene Satz wird in einem string „eingabe“ gespeichert. Nachdem der Benutzer den Satz eingegeben hat wird die Methode „Decrypt.Start“[2] aufgerufen. Diese bekommt die Werte „key“ und „eingabe“ übergeben.

## 1.4 BruteForce

Wenn der „Bruteforce Modus“ gewählt wird, wird der Benutzer nach einem Satz gefragt welcher entschlüsselt werden soll. Der eingegebene Satz wird in eine string Variable „eingabe“ gespeichert. Anschließend wird die Methode „BruteForce.Start“[3] aufgerufen welche den Wert „eingabe“ übergeben bekommt.

## 1.5 Textdatei einlesen und entschlüesseln lassen

Wenn der „Textdatei einlesen und entschlüsseln lassen Modus“ gewählt wird, wird der Benutzer gefragt, um wie viele Stellen er die Nachricht verschieben will. Die eingegebene Zahl wird in einem Integer „key“ gespeichert. Im Anschluss wird der Benutzer nach dem Pfad zu der verschlüsselten Textdatei gefragt. der Pfad wird als String „pfad“ gespeichert. Im Anschluss wird die Methode „TextDecrypt.Start“[4] gestartet und bekommt die Werte „key“ und „pfad“ übergeben.

## 1.6 Textdatei einlesen und verschlüsseln lassen

Wenn der „Textdatei einlesen und verschlüsseln lassen Modus“ gewählt wird, wird der Benutzer gefragt, um wie viele Stellen er die Nachricht verschieben will. Die eingegebene Zahl wird in einem Integer „key“ gespeichert. Im Anschluss wird der Benutzer nach dem Pfad zu der entschlüsselten Textdatei gefragt. der Pfad wird als String „pfad“ gespeichert. Im Anschluss wird die Methode „TextDecrypt.Start“[5] gestartet und bekommt die Werte „key“ und „pfad“ übergeben.

# 2 Encrypt.cs

## 2.1 Start [7]

Die Start Methode bekommt die Werte (int)„key“ und (String) „eingabe“ übergeben. Es wird ein StringBuilder mit dem Namen „neuerSatz“ erschaffen. Anschließend werden die characters des Strings „eingabe“ zu einem byte Array mit dem Namen „asciiByte“ konvertiert. Sollte „key“ größer sein als die Anzahl der Einträge in der Ascii-Tabelle, so wird „key“ mit „key“ % länger der Ascii-Tabelle überschrieben um zu garantieren, dass man nicht „out of bounce“ laufen kann. Im Anschluss daran wird für jeden Eintrag im „asciiBytes“ Array der Ascii-Wert + den „key“-Wert gerechnet und zurück in einen Char konvertiert und zu dem Stringbuilder hinzugefügt. So entsteht nach und nach der entschlüsselte Satz.

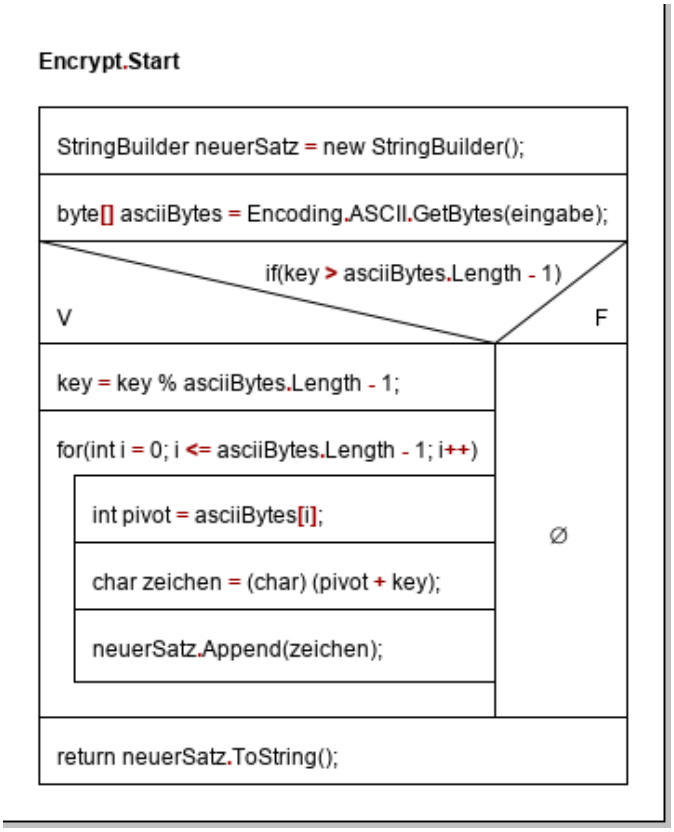



Abbildung 2.1: Encrypt.Start(int key, string eingabe)

# 3 Decrypt.cs

## 3.1 Start [8]

Die Start Methode bekommt die Werte (int)„key“ und (String) „eingabe“ übergeben. Es wird ein StringBuilder mit dem Namen „neuerSatz“ erschaffen. Anschließend werden die characters des Strings „eingabe“ zu einem byte Array mit dem Namen „asciiByte“ konvertiert. Anschließend wird eine Schleife gestartet welche durch den gesammten Satz durch läuft und bei jedem durchlauf erst einen Wert lädt und auf die integer Variable „pivot“ schreibt. Anschließend wird die Methode „CheckLower“[9] gestartet welche die Differenz von „pivot“-„key“ übergeben bekommt. Das Ergebnis daraus wird auf die Variable „key“ geschrieben. Im nächsten Schritt wird das „asciiByte“ wieder zu einem Char konvertiert und zum Stringbuilder hinzugefügt.

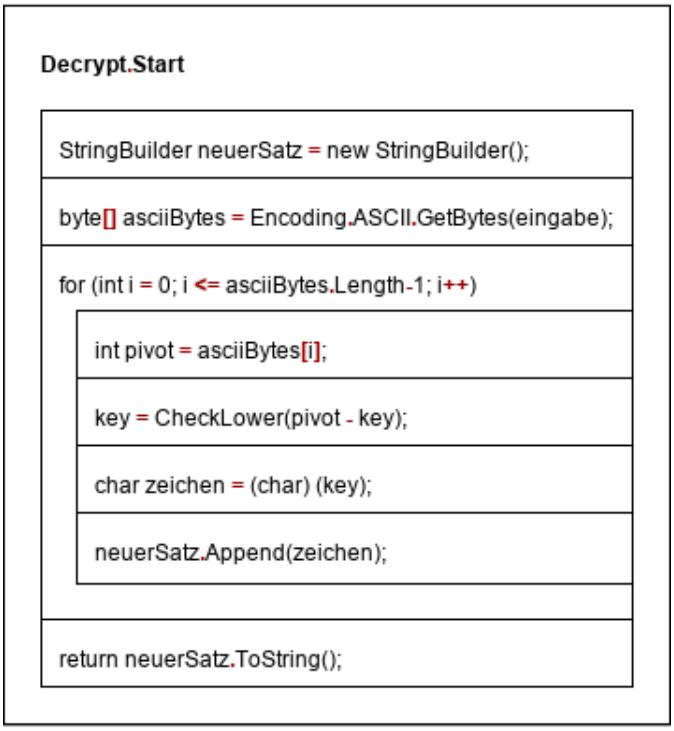



Abbildung 3.1: Decrypt.Start(int key, string eingabe)

## 3.2 CheckLower [9]

Die Methode „CheckLower“ wird aufgerufen und bekommt den integer „asciiChar“ übergeben. Es wird eine if-Verzweigung gestartet welche fragt, ob „asciiChar“ $< 0$ ist um zu verhindern, dass beim Entschlüsseln „Out of Bounce“ ein Problem darstellt. Wenn „asciiChar“ $< 0$ ist, wird 127 auf „asciiChar“ addiert und die Methode mit „asciiChar“ erneut aufgerufen. Wenn „asciiChar“ nicht $< 0$ ist, wird „asciiChar“ zurück gegeben.

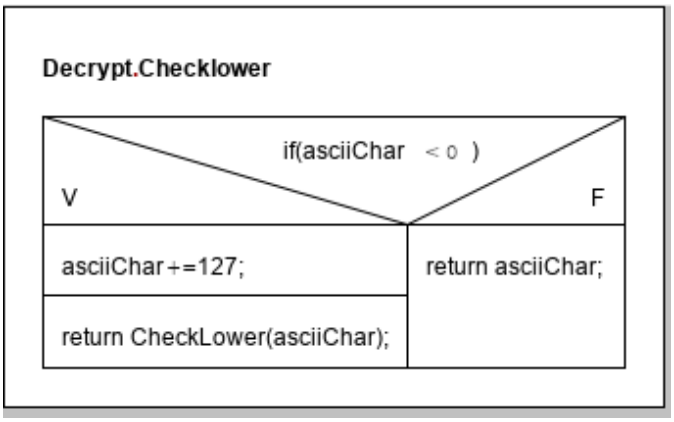



Abbildung 3.2: Decrypt.Start(int asciiChar)

# 4 BruteForce.cs

## 4.1 Start

BruteForce.Start wird aufgerufen und bekommt einen String „text“ übergeben. Als erstes wird ein Integer „asciiLength“= 127 initialisiert. 128 ist der Wert der Ascii-Tabelle von 1963 doch da die Tabelle bei 0 anfängt zu zählen muss 1 von 128 abgezogen werden. Im Anschluss daran wird eine for-Schleife gestartet welche von 0 bis 127 läuft und in jedem Durchlauf die Methode „Decrypt.Start“[8] mit den Werten „key“ und „text“ aufruft.

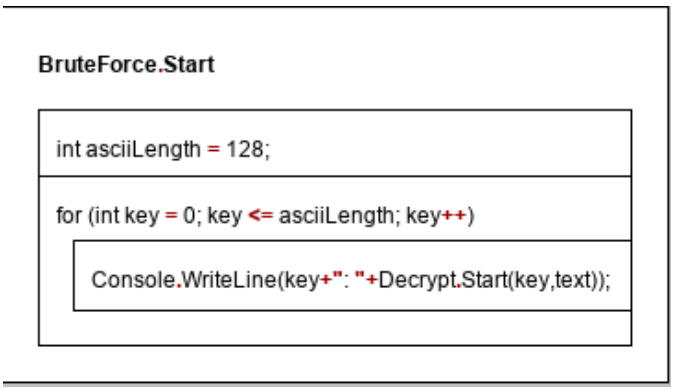



Abbildung 4.1: BruteForce.Start

# 5 TextDecrypt.cs

## 5.1 Start

Die Methode „TextDecrypt.Start“ wird mit den Werten „key“ und „path“ gestartet. Als erstes wird die angegebene Textdatei in der Variable „file“ gespeichert. Anschließend wird ein string „text“ erschaffen, welcher den Inhalt der Textdatei übergeben bekommt. Als nächstes wird eine neue Textdatei erstellt welche im selben Verzeichnis liegt wie die Ursprungsdatei. Die Methode „Decrypt.Start“[8] wird mit den Werten „key“ und „text“ gestartet und in die neue Textdatei geschrieben. Sollte kein Fehler auftreten gibt das Programm auf der Konsole „Fertig“ aus. Wenn ein Fehler auftritt wird der Fehler auf der Konsole ausgegeben und „ERROR“ geschrieben.

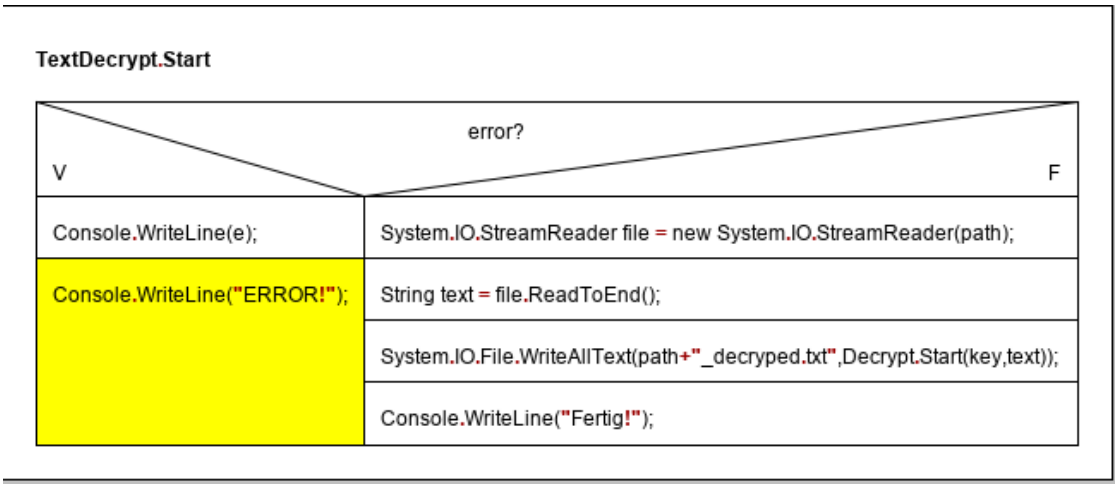



Abbildung 5.1: TextDecrypt.Start

# 6 TextEncrypt.cs

## 6.1 Start

Die Methode „TextEncrypt.Start“ wird mit den Werten „key“ und „path“ gestartet. Als erstes wird die angegebene Textdatei in der Variable „file“ gespeichert. Anschließend wird ein string „text“ erschaffen, welcher den Inhalt der Textdatei übergeben bekommt. Als nächstes wird eine neue Textdatei erstellt welche im selben Verzeichnis liegt wie die Ursprungsdatei. Die Methode „Encrypt.Start“[8] wird mit den Werten „key“ und „text“ gestartet und in die neue Textdatei geschrieben. Sollte kein Fehler auftreten gibt das Programm auf der Konsole „Fertig“ aus. Wenn ein Fehler auftritt wird der Fehler auf der Konsole ausgegeben und „ERROR“ geschrieben.

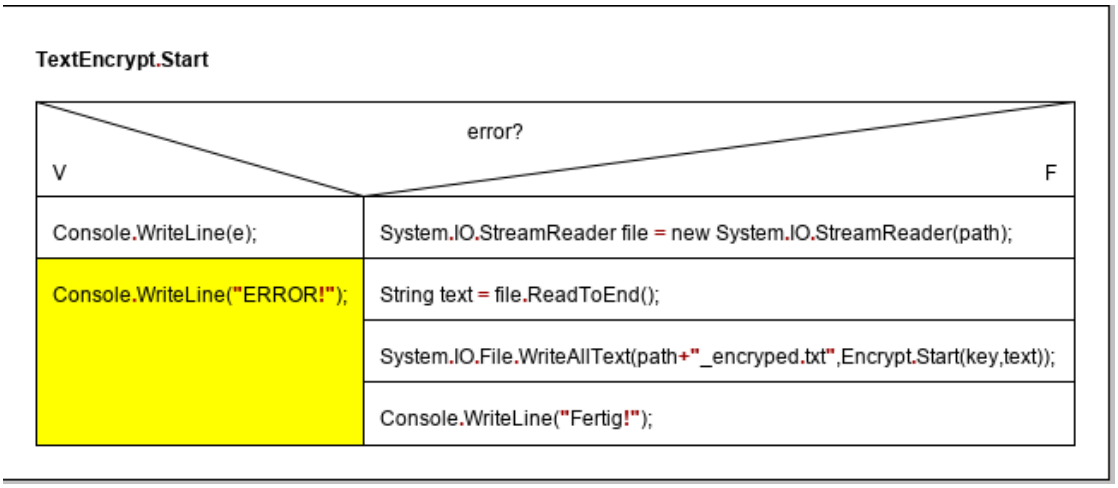



Abbildung 6.1: TextEncrypt.Start