Brainfuck für den Commodore 64

Vintage Computing Carinthia
Dokumentation von Wil

Version 1.0
April 2022

Inhaltsverzeichnis

1 Brainfuck - Was, Warum und Wie 2

2 Die Brainfuck-Programmiersprache 2
- 2.1 Speicherstruktur . . . 2
- 2.2 Die acht Befehle . . . 2
- 2.3 Rechnen in Brainfuck . . . 3
- 2.4 Text ausgeben . . . 4
- 2.5 Programmsteuerungsablauf . . . 5

3 Brainfuck auf dem Commodore 64 7
- 3.1 Der Editor . . . 7
- 3.2 Ein erstes Brainfuck-Programm . . . 8
- 3.3 Speichern und Laden von Programmen . . . 9
- 3.4 Zeichenkodierung . . . 10
- 3.5 Debugging . . . 10

4 Codebeispiele auf der Diskette 11
- 4.1 Hello World . . . 11
- 4.2 Echo . . . 11
- 4.3 Brain - eine PETSCII-Grafik . . . 12
- 4.4 e - Eulersche Zahl . . . 12
- 4.5 Life - Game of Life . . . 12
- 4.6 TTT - Tic-Tac-Toe . . . 13
- 4.7 Clocktower - Grazer Uhrturm . . . 13
- 4.8 Primes - Primzahlen berechnen . . . 13
- 4.9 Fantasie - Ein Gedicht gegen den Krieg . . . 14

1 Brainfuck - Was, Warum und Wie

Brainfuck ist eine Computerprogrammiersprache, die 1993 von Urban Müller eingeführt wurde. Auf den ersten Blick scheint sie die Langsamkeit von BASIC mit der Schwierigkeit der Assembler-Programmierung zu verbinden, aber tatsächlich übertrifft sie beide Sprachen in diesen Aspekten. Sie ist viel langsamer als BASIC und viel schwieriger als 6502 Assembly. Also, willkommen bei Brainfuck!

Dies ist die Dokumentation für Brainfuck64 V1.0 von Vintage Computing Carinthia. Es ist nicht die erste Implementierung eines Brainfuck-Interpreters, es gab zumindest tolle Implementierungen von TheDreams in 2001[1] und von Marco64 in 2009[2]. Aber jetzt haben Sie mit Brainfuck64 ein Brainfuck Entwicklungssystem für Ihren Commodore 64 erworben!

2 Die Brainfuck-Programmiersprache

2.1 Speicherstruktur

Ein Brainfuck-Programm besteht aus einer beliebigen Kombination von Brainfuck-Befehlen und darüber hinaus aus weiterem Text, der bei der Ausführung des Programms ignoriert wird. Auf diese Weise können Kommentare direkt in den Code geschrieben werden, solange sie keine Brainfuck-Befehle enthalten. Ein Wert von 0 markiert das Ende des Codes.

Auf den Arbeitsspeicher wird durch ein laufendes Brainfuck-Programm zugegriffen und dieser manipuliert. Das Programm kann immer nur auf eine Speicherzelle zugreifen, was durch einen Zeiger auf die aktuelle Speicherzelle angezeigt wird. Der Arbeitsspeicher wird bei jedem Programmstart mit Nullen initialisiert und der Zeiger zeigt zunächst auf die erste Zelle.

2.2 Die acht Befehle

Die Brainfuck-Sprache hat nur acht Befehle, ja acht. Jeder Befehl wird durch ein einzelnes Zeichen dargestellt. Die acht Befehle sind:

```
.  , + - > < [ ]
```

. gibt das Zeichen mit dem ASCII-Wert in der aktuellen Speicherzelle aus.

, liest ein Zeichen von der Tastatur und setzt den ASCII-Code in die aktuelle Speicherzelle.

---

1. Brainf... https://csdb.dk/release/?id=74814
2. Brainfuck Interpreter https://csdb.dk/release/?id=74303

+ erhöht den Wert in der aktuellen Speicherzelle um 1. Wenn der maximale Zellenwert (typischerweise 255) überschritten wird, wird der Wert auf 0 gesetzt.

- verringert den Wert in der aktuellen Speicherzelle um 1. Wenn der Zellwert bereits seinen Minimalwert (0) enthält, wird der Wert auf den maximalen Zellwert (typischerweise 255) umgebrochen.

> erhöht den Zeiger auf die aktuelle Speicherzelle um 1

< erniedrigt den Zeiger auf die aktuelle Speicherzelle um 1

[wenn die aktuelle Speicherzelle 0 enthält, springt der Programmablauf hintern den zugehörigen]-Befehl vorwärts. Wenn die aktuelle Speicherzelle eine andere Zahl als 0 enthält, fährt das Programm mit dem Befehl direkt nach dem [fort.

] Wenn die aktuelle Speicherzelle eine andere Zahl als 0 enthält, springt der Programmfluss auf den Befehl nach dem zugehörigen [zurück. Wenn die aktuelle Speicherzelle 0 enthält, fährt das Programm mit dem Befehl direkt nach dem] fort.

2.3 Rechnen in Brainfuck

2.3.1 Setzen eines Wertes

Setze aktuelle Speicherzelle auf 0: [-] führt eine Schleife aus die den Wert der aktuellen Speicherzelle solange erniedrigt bis er 0 ist. Alle Arbeitsspeicherzellen werden mit 0 initialisiert, so dass dieses Codestück nur notwendig ist, wenn eine Zelle zuvor für eine Berechnung verwendet wurde.

Setze aktuelle Speicherzelle auf X: Wenn die Speicherzelle zuvor 0 war, erledigt eine Folge von X +-Befehlen den Trick. Zum Beispiel setzt +++++ einen Wert von 5.

Setze aktuelle Speicherzelle auf to X (kürzerer Programmcode): Größere Zahlen können auf raffiniertere Weise mit Schleifen definiert werden. So ergibt beispielsweise +++++[>+++++++++++++<-]> eine Zelle mit dem Wert 65, der sich aus der Addition von 5 und 13 ergibt. Dies verbraucht einen zusätzlichen Speicherplatz, ist aber viel kürzer als 65 +-Befehle in einer Reihe. Eine noch kürzere Definition findet man, wenn man das Wrapping von Werten verwendet: ----[---->+<]>++ zählt bis 63, während es in 4 Schritten abwärts zählt. Nach der Schleife wird der Wert zweimal erhöht, um wieder 65 zu ergeben.

Für eine Liste der Konstanten, die in kompaktem Brainfuck-Code ausgedrückt werden, siehe Brainfuck-Konstanten [3].

2.3.2 Verschieben und Duplizieren eines Wertes

Das Verschieben eines Wertes erfolgt durch Dekrementieren des ursprünglichen Wertes und gleichzeitigem Inkrementieren des neuen Wertes: [>+<-] Die Zielspeicherzelle muss 0 sein (oder auf 0 gesetzt werden), damit dieser Algorithmus funktioniert.

[>+>+<<-] kopiert den Wert der aktuellen Speicherzelle in die nächsten beiden Zellen, die ursprüngliche Zelle wird dabei auf 0 gesetzt. Um den ursprünglichen Wert in der ersten Zelle zu setzen, muss ein Wert zurückgeschoben werden, indem dieser Code hinzugefügt wird: >>[<<+>>-] Nach der Schleife zeigt der Speicherzeiger auf die 0 nach den beiden mit dem Wert gefüllten Zellen.

2.3.3 Addieren und Subtrahieren von Werten

[>+<-] addiert den Wert der aktuellen Zelle zum Wert der nächsten Zelle.

[>-<-] subtrahiert den Wert der aktuellen Zelle vom Wert der nächsten Zelle.

2.3.4 Multiplizieren von Werten

Die Multiplikation des aktuellen Zellenwerts mit einer Konstante X funktioniert, indem X wiederholt zum Ergebnis addiert wird, während der erste Zellenwert dekrementiert wird: [>(X mal +)<-]

Multiplikation zweier Zellwerte [>[>+>+<<-]>>[<<+>>-]<<<-] durch iterative Addition.

2.3.5 Division

Wenn die Division einen Rest von 0 hat, funktioniert die folgende Schleife, um durch X zu dividieren [>+<(X mal -)]

Die Division von zwei Zellwerten mit einem möglichen Rest erfordert etwas mehr Aufwand: [->>+<-[>>>]>[[<+>-]>+>>]<<<<<] dividiert den aktuellen Zellwert durch den nächsten Zellwert und trägt Rest und Divisionsergebnis in die beiden aufeinanderfolgenden Zellen ein.

2.4 Text ausgeben

Um ein Textzeichen auszugeben, setzen Sie den entsprechenden ASCII-Wert in eine Speicherzelle und geben ihn mit . aus.

Da . den Wert der Zelle nicht verändert, kann der Wert der vorherigen Ausgabe zur Erzeugung des nächsten Wertes verwendet

3. Brainfuck constants - Esolang https://esolangs.org/wiki/Brainfuck_constants

werden. So kann der Name einer großen Band [4] auf sehr kompakte Weise ausgedrückt werden:

```
----[---->+<]>++.+..-.
```

Die Ausgabe des Wertes 10 erzeugt einen Wagenrücklauf.

2.5 Programmsteuerungsablauf

2.5.1 Wiederhole X-mal

Dies ist eine der häufigsten Strukturen in Brainfuck, wir haben sie unter anderem schon für den Multiplikationsalgorithmus verwendet.

(X mal +)[Code in der Schleife -]

Das folgende Beispiel gibt 10 Ausrufezeichen aus:

```
++++++++++[> ++++[->++++++++<]>+.[-]<[-] <-]
```

Der ASCII-Wert des Ausrufezeichens wird mit Hilfe von zwei Speicherzellen erzeugt. Die Sequenz [-]<[-] nach der Ausgabe löscht die Werte für den nächsten Durchlauf und stellt den Zellenzeiger ein.

2.5.2 Wenn nicht Null

Der Befehl [erledigt bereits den größten Teil der Arbeit. Um die Schleife zu verlassen, muss an der aktuellen Zelle beim Erreichen der]-Zeile ein Nullwert erscheinen. Wenn der ursprüngliche Wert danach nicht mehr benötigt wird, erledigt ein [-] die Aufgabe:

[Code, der bedingt ausgeführt wird [-]]

Wenn bekannt ist, dass die nächste Zelle leer ist, kann das [-] weggelassen werden, wodurch der getestete Wert erhalten bleibt.

[bedingt auszuführender Code >]<

2.5.3 Wenn Null

Um auf Null zu testen, setzen wir ein Flag in der nächsten Zelle auf 1 und testen dann, ob der Wert der ersten Zelle ungleich Null ist. Wenn dies der Fall ist, wird das Kennzeichen zurückgesetzt. Mit anderen Worten: Das Flag wird als logisches NICHT des aktuellen Zellwerts berechnet.

>+<[>-<[-]]>[Code, der bedingt ausgeführt wird -]

In Kombination mit der Subtraktionsmethode kann das Wenn-Null-Muster auch verwendet werden, um eine "Wenn gleicher Wert"-Bedingung zu formulieren.

---

4. Vielleicht haben Sie ----[---->+<]>++.++.+.-.

2.5.4 Wenn ... dann ... sonst ...

Der folgende Code implementiert eine Wenn-0-dann-sonst-Struktur.

```
   >+<[>-< hier kommt der "sonst"-Code[-]]>[- hier kommt der
"dann"-Code [-]]
```

Aufgrund der verschachtelten Schleife steht der sonst-Teil im Code an erster Stelle. Der sonst-Code muss so enden, dass die nächste Speicherzelle 0 ist.

2.5.5 Entscheiden anhand verschiedener Fälle

Dies kann auf ähnliche Weise implementiert werden wie der wenn-dann-sonst-Ansatz. Zum Beispiel:

```
Entscheide anhand von (X)
Fall 3:
 1 ausgeben
Fall 8:
 2 ausgeben
Fall 6:
 3 ausgeben
Alle anderen Fälle:
 ? ausgeben
```

Für einen effizienten Vergleich sollten die Werte bei den Fällen von niedrig nach hoch sortiert werden:

```
Entscheide anhand von (X)
Fall 3:
 1 ausgeben
Fall 6:
 3 ausgeben
Fall 8:
 2 ausgeben
Alle anderen Fälle:
 ? ausgeben
```

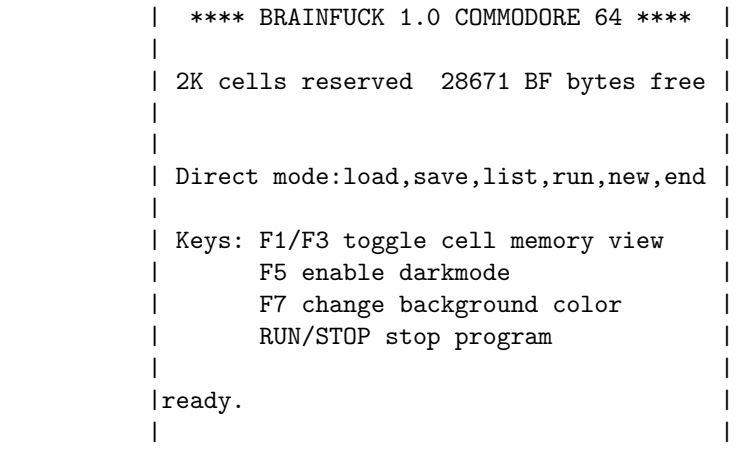
Der entsprechende Brainfuck-Code lautet dann wie folgt:

```
>+< Flagge setzen
---[ vergleiche mit 3
---[ vergleiche mit 6
---[ vergleiche mit 8
>-< Standardteil
[-]----[---->+<]>.[-]< ? ausgeben und verwendete Zelle löschen
[-]] Schleife verlassen
>[- Fall 8
-[----->+<]>--.[-]< [-] 3 ausgeben und Speicherzellen löschen
]<] end case
>[- Fall 6
-[----->+<]>-.[-]< [-] 2 ausgeben und Speicherzellen löschen
```

```
]<] end case
>[- Fall 3
-[----->+<]>.[-]< [-] 1 ausgeben und Speicherzellen löschen
] end case und switch
```

3 Brainfuck auf dem Commodore 64

Nach dem Laden und Starten von brainfuck64 begrüßt Sie der Computer mit der folgenden Meldung:



Es gibt maximal 2048 Speicherzellen. Jede Zelle ist als 8-Bit-Ganzzahl implementiert, die auch überlaufen kann. Wenn also die aktuelle Zelle den Wert 255 hat, wird sie mit dem Befehl + auf 0 gesetzt. Zusätzlich zu den Speicherzellen stehen 28 KB für Brainfuck-Programme zur Verfügung.

3.1 Der Editor

Der Editor ist ein Vollbild-Editor, der Zeilennummern zum Hinzufügen, Löschen oder Ändern des Programms verwendet.

Um eine Programmzeile hinzuzufügen, geben Sie einfach eine Zeilennummer ein, gefolgt von dem Code, der unter dieser Nummer gespeichert werden soll, und drücken Sie die Eingabetaste. Die maximale Länge einer Zeile beträgt 80 Zeichen (zwei Bildschirmzeilen). Wenn diese Zeilennummer bereits in Ihrem Code vorhanden ist, wird die Zeile durch die neue Zeile ersetzt. Um eine Zeile zu löschen, geben Sie die einfache Zeilennummer ein und drücken Sie die Eingabetaste. Um das gesamte Programm zu löschen, geben Sie NEW ein und drücken Sie die Eingabetaste.

Zeilennummern können eine beliebige ganze Zahl zwischen 0 und 63999 sein. Es wird empfohlen, Zeilennummern mit Abstand zu vergeben, um Platz für spätere Ergänzungen zu lassen. Eine gängige Praxis ist es, einen Abstand von 10 zu verwenden, zum Beispiel 10, 20, 30, 40, 50, ...

Der Computer sortiert Ihre Zeilen entsprechend ihrer Nummer in ein Programm. Um Ihr Programm zu sehen, geben Sie LIST ein.

Für den Fall, dass Ihr Programm länger als ein Bildschirm ist, kann LIST zusammen mit einer Zeilennummer oder einem Bereich von Zeilennummern verwendet werden:

```
LIST          Zeigt das gesamte Programm an
LIST 100      Zeigt nur Zeile 100
LIST -100     Listet das Programm von Anfang bis Zeile 100
LIST 100-     Listet das Programm von Zeile 100 bis zum Ende des Programms auf
LIST 100-200  Listen des Programms von Zeile 100 bis Zeile 200
```

Bei der Ausführung eines Programms macht es keinen Unterschied, ob ein Brainfuck-Programm über mehrere Zeilen verteilt ist oder nicht. Um ein Programm zu starten, geben Sie RUN ein und drücken die Eingabetaste. RUN kann auch mit einer Zeilennummer verwendet werden, um das Programm an der angegebenen Nummer zu starten. Während der Ausführung des Programms wird der Rahmen grau. Nach Beendigung des Programms wechselt die Farbe des Rahmens wieder zurück. Ein laufendes Programm kann auch durch Drücken der RUN/STOP-Taste angehalten werden.

Um Brainfuck zu verlassen und zu BASIC zurückzukehren, geben Sie END ein und drücken Sie die Eingabetaste. Warnung: ein Programm im Speicher wird in diesem Fall gelöscht, also stellen Sie sicher, dass Sie Ihr Programm vorher speichern.

3.2 Ein erstes Brainfuck-Programm

Beginnen Sie mit einem leeren Programm und geben Sie die folgenden Zeilen ein:

```
10 -[------->+<]>-.[-]
20 --[----->+<]>-.[-]
30 +[------->++<]>--.[-]
40 +[------->++<]>--.[-]
50 +[------->++<]>+.
```

Geben Sie LIST ein, um zu prüfen, ob alle Zeilen vorhanden sind.

Tippen Sie RUN und prüfen Sie die Ausgabe.

Geben Sie RUN 20 ein und sehen Sie nach, was anders ist.

Geben Sie 30 ein, drücken Sie die Eingabetaste und führen Sie erneut RUN aus. Was ist passiert?

Tippen Sie LIST 40 ein, bewegen Sie den Cursor auf die Zeilennummer und ändern Sie sie in 30, dann drücken Sie die Eingabetaste. Tippen Sie LIST. Die Zeile 40 ist nun in die Zeile 30 kopiert worden.

Beachten Sie, dass die Funktionalität des Beispielprogramms auch kompakter geschrieben werden kann, indem der Wert aus der vorherigen Berechnung für alle Zeichen außer dem ersten genutzt wird.

```
1 -[------->+<]>-.-[->+++++<]>++.
2 +++++++..+++.
```

Die Ausführung der kompakten Version ist auch sichtbar schneller.

3.3 Speichern und Laden von Programmen

3.3.1 Speichern eines Programms

Brainfuck-Programme werden mit den gleichen Befehlen wie in BASIC gespeichert. Um ein Programm zu speichern, tippen Sie

```
SAVE "Programmname",X
```

wobei X die Adresse Ihres peripheren Speichergeräts ist. Bei einem Diskettenlaufwerk ist das normalerweise die Nummer 8.

Wenn bereits eine Datei mit demselben Namen existiert, schlägt der Befehl SAVE fehl. Es gibt eine Möglichkeit, eine bestehende Datei zu überschreiben, aber beachten Sie einen Fehler im 1541-Laufwerk, der dazu führt, dass Daten beim überschreiben verloren gehen können. Um diesen Fehler zu vermeiden, sollten Sie die optionale Laufwerksnummer vor dem Dateinamen angeben:

```
SAVE "@0:Name",8
```

3.3.2 Laden eines Programms

Um ein Programm zu laden, geben Sie folgendes ein

```
LOAD "Programmname",X
```

wobei X die Adresse Ihres peripheren Speichergeräts ist. Bei einem Diskettenlaufwerk ist das normalerweise die Nummer 8. Der Programmname kann auch Platzhalter enthalten, wobei ? ein beliebiges Zeichen ersetzt und * für eine beliebige Anzahl von Zeichen steht. Wenn ein mit Platzhaltern angegebener Dateiname mit mehreren Dateien auf einer Festplatte übereinstimmt, wird die erste passende Datei geladen.

3.3.3 Das Verzeichnis der Festplatte anzeigen

Festplattenlaufwerke erlauben das Laden einer speziellen Verzeichnisdatei namens $, die eine Liste aller Dateien auf der Festplatte enthält. Vorsicht, das Laden des Verzeichnisses ersetzt das aktuelle Programm im Speicher. Nach dem Laden mit

```
LOAD"$",8
```

kann das Verzeichnis mit LIST eingesehen werden.

3.4 Zeichenkodierung

Der Commodore C64 hat eine vom Standard-ASCII abweichende Implementierung, den PET Standard Code of Information Interchange (PETSCII). Um Kompatibilität zu gewährleisten, konvertiert Brainfuck64 die ASCII-Codes in ihre jeweiligen PETSCII-Darstellungen. So wird eine Zahl von 65 ein großes A ergeben, wie es in ASCII vorgesehen ist, und der Wagenrücklaufcode ist 10.

Neben den ASCII-Standardcodes interpretiert Brainfuck64 auch PETSCII-Steuercodes und Grafikzeichen. So ist es möglich, den Bildschirm zu löschen, die Farbe des Cursors zu ändern oder grafische Zeichen auf den Bildschirm zu bringen.

Einige Codes sind in der folgenden Tabelle aufgeführt:

19	Cursor-Homeposition
147	Bildschirm löschen
135	F5-Taste (schwarzen Hintergrund einstellen)
136	Taste F7 (nächste Hintergrundfarbe)
17	Cursor nach unten bewegen
157	Cursor nach links bewegen
29	Cursor nach rechts bewegen
145	Cursor nach oben bewegen
18	Umkehrmodus einschalten
144	Schwarze Cursorfarbe einstellen
31	Blaue Cursorfarbe einstellen
149	Braune Cursorfarbe einstellen
159	Cyanfarbene Cursorfarbe einstellen
151	Dunkelgraue Cursorfarbe einstellen
30	Grüne Cursorfarbe einstellen
154	Hellblaue Cursorfarbe einstellen
155	Hellgraue Cursorfarbe einstellen
153	Hellgrüne Cursorfarbe einstellen
150	Hellrote Cursorfarbe einstellen
152	Farbe des mittelgrauen Cursors einstellen
129	Orange Cursorfarbe einstellen
156	Lila Cursorfarbe einstellen
28	Rote Cursorfarbe einstellen
5	Weiße Cursorfarbe einstellen
158	Gelbe Cursorfarbe einstellen
32	Leerzeichen
14	Zeichensatz auf Groß-/Kleinschreibung umstellen
142	Zeichensatz auf Großbuchstaben/Grafikmodus umstellen

3.5 Debugging

3.5.1 Ansicht des Arbeitsspeichers

Brainfuck64 unterstützt eine Funktion zum Betrachten des Arbeitsspeichers während der Ausführung. Drücken Sie F1, um in die Arbeitsspeicheransicht zu wechseln. Dies zeigt die

ersten 1000 Bytes des Arbeitsspeichers mit Life-Updates an. Der Speicherinhalt wird mit hexadezimalen Zahlen dargestellt. Die zweite Speicherseite kann durch erneutes Drücken von F1 angezeigt werden. Die Farbe des Rahmens ändert sich von schwarz zu dunkelgrau, wenn die zweite Seite angezeigt wird. Um die Speicheransicht zu verlassen, drücken Sie F3.

3.5.2 Ausführung anhalten

Um zu sehen, ob der Code eine bestimmte Zeile erreicht und dort anzuhalten, um den Speicherstatus mit F1 zu überprüfen, kann der folgende Code verwendet werden:

```
>>>>>----[>---<--]>++.[-]<[-],[-]<<<<<
```

Dies funktioniert, solange der Speicher 5 Zellen weiter vorne nicht verwendet wird, andernfalls muss die Anzahl von > und < erhöht werden. Es wird empfohlen, diesen Code in einer separaten Zeile zu speichern und diese Zeile zwischen Ihre Programmzeilen zu kopieren, wenn Sie sie zum Anhalten benötigen. Der Code ändert den Hintergrund und fordert zu einer Eingabe auf, dann wird der Speicherzeiger wieder an die Stelle zurückgesetzt, an der er vorher war.

3.5.3 Auskommentieren von Codeteilen

Lassen Sie den Speicherzeiger auf eine Zelle mit 0 zeigen und setzen Sie ein [und] um den auszukommentierenden Code. Dann überspringt der Interpreter diesen Teil.

```
[-][ dieser Code wird nie ausgeführt ]
```

Beachten Sie, dass dies den Nebeneffekt hat, dass die aktuelle Speicherzelle auf 0 gesetzt wird.

Alternativ dazu wird eine [Code] Struktur nach einem] garantiert übersprungen, da die aktuelle Speicherzelle nach Verlassen einer [ ] Schleife 0 sein muss.

4 Codebeispiele auf der Diskette

4.1 Hello World

Das Programm berechnet die ASCII-Werte für die Zeichenkette "Hello World!" und gibt sie auf dem Bildschirm aus.

4.2 Echo

Das Programm liest von der Tastatur und gibt die Eingabe auf dem Bildschirm aus. Das Programm wird durch Drücken der Return-Taste beendet.

4.3 Brain - eine PETSCII-Grafik

Dieses Programm erzeugt die grafische Ausgabe eines Gehirns als PETSCII-Grafik mit einigen Graffiti darunter, gezeichnet von Logiker. Das Programm gibt am Anfang ein Zeichen mit dem Code 142 aus, um den Zeichensatz auf einem Commodore-Computer umzuschalten. Daher würde die Ausgabe dieses Programms auf anderen Brainfuck-Systemen anders aussehen.

4.4 e - Eulersche Zahl

Das Programm berechnet die Eulersche Zahl, Ziffer für Ziffer. Wenn Sie diese Zahl nicht kennen, probieren Sie das Programm einfach aus! Dieses Programm wurde von Daniel Cristofani von brainfuck.org geschrieben und unter einer Creative Commons BY-SA 4.0 Lizenz veröffentlicht.

4.5 Life - Game of Life

Das Programm simuliert ein Game of Life System innerhalb eines 10x10 Rasters. Game of Life ist ein zweidimensionaler zellulärer Automat, der von John Conway erfunden wurde. Jede Zelle kann einen toten oder einen lebendigen Zustand haben. Die acht Zellen um eine Zelle herum bilden die Nachbarschaft einer Zelle. In jedem Simulationsschritt ändern die Zellen ihren Zustand nach den folgenden Regeln:

- eine lebende Zelle mit weniger als 3 Nachbarn stirbt an Einsamkeit
- eine tote Zelle mit genau 3 lebenden Nachbarn wird geboren (wechselt in den lebenden Zustand)
- eine lebende Zelle mit mehr als 3 Nachbarn stirbt an Überbevölkerung
- alle anderen Zellen behalten ihren Zustand

Um das Programm zu benutzen, starten Sie es und warten Sie, bis das (leere) Raster gezeichnet ist. An der Eingabeaufforderung können Sie entweder die Koordinaten einer Zelle eingeben oder die Eingabetaste drücken, um die Simulation um einen Schritt voranzutreiben.

Einige Muster, mit denen Sie beginnen können:

```
**          *          *
**          *           *
            *         ***
```

Der Block ist ein statisches Muster, der Balken oszilliert und das Muster auf der rechten Seite ist ein Gleiter, der sich diagonal über den Bildschirm bewegt, indem er in zwei Schritten eine verschobene Kopie von sich selbst erstellt.

Dieses Programm wurde von Daniel Cristofani von brainfuck.org geschrieben und unter einer Creative Commons BY-SA 4.0 Lizenz veröffentlicht. Wir haben ein . nach dem , in Zeile 235 hinzugefügt, damit die Eingabe auf dem Bildschirm angezeigt wird.

4.6 TTT - Tic-Tac-Toe

Dieses Programm spielt Tic-Tac-Toe. Nach dem Start macht das Programm seinen ersten Zug. Der Spieler kann daraufhin reagieren, indem er ein verbleibendes Feld auswählt. Da Game of Life eigentlich eine Simulation ist, ist dies das einzige Spiel in unseren Beispielen.

Das Spiel führt viele Berechnungen durch und braucht daher sehr lange, um zu reagieren, besonders beim ersten Mal nachdem der Spieler seinen Zug eingegeben hat. Wenn Sie das Warten satt haben, drücken Sie die RUN/STOP-Taste, um das Programm anzuhalten.

Dieses Programm wurde von Daniel Cristofani von brainfuck.org geschrieben und unter einer Creative Commons BY-SA 4.0 Lizenz veröffentlicht. Wir haben ein . nach dem , zweimal in Zeile 15 eingefügt, damit die Eingabe auf dem Bildschirm angezeigt wird.

4.7 Clocktower - Grazer Uhrturm

Dieses Programm schaltet in den Grafikmodus und gibt einen F5-Tastencode aus, um den Hintergrund auf Schwarz zu schalten. Dann wird eine bildschirmfüllende Grafik des Grazer Uhrturms bei Nacht gedruckt. Grüße an Commodore Treffen Graz an dieser Stelle! Die Ausgabe erfolgt mit den PETSCII-Grafikzeichen und Codes zur Änderung der Cursorfarbe. Daher würde die Ausgabe dieses Programms auf anderen Brainfuck-Systemen ganz anders aussehen.

4.8 Primes - Primzahlen berechnen

Dieses Programm gibt die Primzahlen bis zu einer bestimmten Zahl aus. Nach dem Start des Programms fragt es nach "Primzahlen bis". Nach Eingabe einer Zahl werden die Primzahlen bis zur eingegebenen Zahl berechnet und ausgedruckt. Bei der Eingabe einer Zahl größer als 255 wird die Zahl bei 8-Bit abgeschnitten, d.h. die Eingabe von 258 ist gleichbedeutend mit der Eingabe von 3.

Dieses Programm wurde von Daniel Cristofani von brainfuck.org geschrieben und unter einer Creative Commons BY-SA 4.0 Lizenz veröffentlicht. Wir haben ein . nach dem , in Zeile 25 hinzugefügt, damit die Eingabe auf dem Bildschirm angezeigt wird.

4.9 Fantasie - Ein Gedicht gegen den Krieg

Fantasie von Übermorgen ist ein Gedicht des deutschen Dichters Erich Kästner, das sich gegen den Krieg wendet und dafür plädiert, dass Frauen ihren Männern nicht erlauben sollten, in den Krieg zu ziehen. Kästner veröffentlichte das Gedicht 1929, zwischen den beiden Weltkriegen. Zum Zeitpunkt der Veröffentlichung dieser Version von Brainfuck64 gibt es immer noch Kriege, sogar in Europa. Das ist schrecklich und unglaublich! Wir haben dieses Gedicht hinzugefügt, um unsere Hoffnung auszudrücken, dass der Krieg irgendwann ein Ende hat. Überall und für immer.