Brainfuck

Lösungen

Lösung 1. Eine mögliche Lösung:

$\underline{0}$	0	0	0	...	
$\underline{1}$	0	0	0	...	+
1	$\underline{0}$	0	0	...	>
1	$\underline{1}$	0	0	...	+
1	$\underline{2}$	0	0	...	+
1	2	$\underline{0}$	0	...	>
1	2	$\underline{1}$	0	...	+
1	2	$\underline{2}$	0	...	+
1	2	$\underline{3}$	0	...	+
1	2	3	$\underline{0}$	...	>

Die entsprechenden Befehle wurden zusätzlich in der rechten Seite vermerkt.

Lösung 2. Eine mögliche Lösung:

$\underline{0}$	0	0	0	...	
$\underline{1}$	0	0	0	...	+
$\underline{2}$	0	0	0	...	[+]
$\underline{3}$	0	0	0	...	[+]
$\underline{4}$	0	0	0	...	[+]
$\underline{5}$	0	0	0	...	[+]

...

d.h. das Programm gerät in eine Endlosschleife.

Lösung 3. Eine mögliche Lösung:

```
,[.+++++++++++++++++++++++++++++++.,]
```

Lösung 4. Das Programm implementiert eine sogenannte Caesar-Shift-Verschlüsselung, bei der jeder Buchstabe um eine Stelle „nach rechts“ verschoben wird.

Für den zweiten Teil der Aufgabe muss das erste Zeichen um eine Stelle, das zweite Zeichen um zwei Stellen, das dritte um drei Stellen und das vierte Zeichen um vier Stellen verschoben werden. Dies kann z.B. wie folgt gelöst werden:

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

Lösung 5. Eine mögliche Lösung:

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

Lösung 6. Der Trick besteht darin, dass man zuerst den Wert des ersten Feldes in die folgenden zwei Felder (ähnlich wie bei der Addition) verschiebt, also aus

a	0	0	0	...

wird

0	a	a	0	...

und dann den Wert des dritten Feldes ins erste zurückkopiert, also

a	a	0	0	...

Dies kann man z.B. wie folgt lösen:

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

(In der ersten Zeile wird der zu kopierende Wert eingeben, in diesem Fall also 4)

Lösung 7. Ähnlich wie bei der Addition kann man die Multiplikation wie folgt auffassen

$$a \cdot b = \underbrace{b + b + \cdots + b}_{a\text{-mal}},$$

also z.B.

$$3 \cdot 5 = 5 + 5 + 5.$$

Wir benötigen also eine zweite Schleife, innerhalb dieser wird die Addition dann genügend oft ausgeführt. Damit wir immer die richtigen Additionswerte zur Verfügung haben, muss dieser zuerst in ein entsprechendes Feld kopiert werden.

Eine mögliche Lösung[1] sieht wie folgt aus:

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

Lösung 8. Eine mögliche Lösung:

[1] von `http://de.wikipedia.org/wiki/Brainfuck`

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

Lösung 9. Eine mögliche Lösung:

```
,[>,]<[.<]
```

Lösung 10. Das erste Programm gerät in eine Endlosschleife, es füllt jedes Band auf dem Feld mit einer Eins, also

0	0	0	0	...
1	0	0	0	...
1	1	0	0	...
1	1	1	0	...
1	1	1	1	...

...

Das zweite Programm durchläuft die Schleife genau einmal und bleibt im ersten Feld, also

0	0	0	0	...
1	0	0	0	...
2	0	0	0	...

Lösung 11. Eine mögliche Lösung:

```java
class MyBrainfuckProgram {
    static final int FIELD_SIZE = 1000;

    public static void main(String[] args) {
        int[] field = new int[FIELD_SIZE];
        int currentPosition = 0;

        field[currentPosition]++;
        field[currentPosition]++;
        field[currentPosition]++;
```

```
        currentPosition++;
        field[currentPosition]++;
        field[currentPosition]++;
        field[currentPosition]++;
        field[currentPosition]++;
        field[currentPosition]++;

        while(field[currentPosition]!=0) {
            currentPosition--;
            field[currentPosition]++;
            currentPosition++;
            field[currentPosition]--;
        }

    }
}
```

Die folgende Lösung läuft online unter `http://clab2.phbern.ch:81/lego/Communication.html`

```
/* Runs in http://clab2.phbern.ch:81/lego/Communication.html */

import ch.aplu.util.*;

class MyBrainfuckProgramApluVersion extends Console {
    static final int FIELD_SIZE = 1000;

    public static void main(String[] args) {
        int[] field = new int[FIELD_SIZE];
        int currentPosition = 0;

        field[currentPosition]++;
        field[currentPosition]++;
        field[currentPosition]++;
        currentPosition++;
        field[currentPosition]++;
        field[currentPosition]++;
        field[currentPosition]++;
        field[currentPosition]++;
        field[currentPosition]++;

        while(field[currentPosition]!=0) {
            currentPosition--;
            field[currentPosition]++;
            currentPosition++;
            field[currentPosition]--;
        }

        print("Der Wert im ersten Feld ist "+field[0]);
```

```
        }
}
```