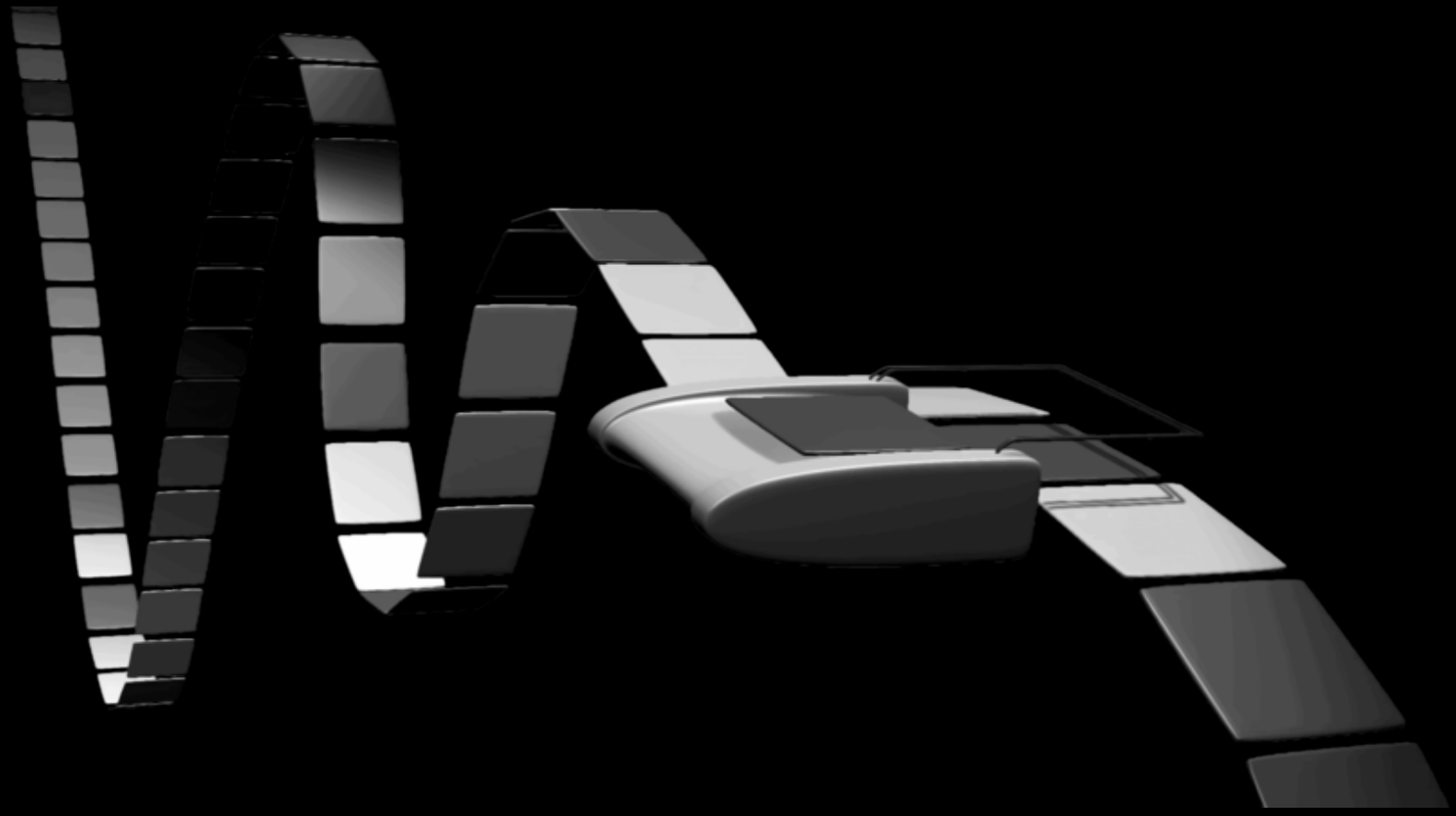
How to program in Brainfuck

Diff

튜링 기계



튜링 기계

- 하나의 무한히 긴 테이프 위를 움직이는 기계.
- 기계는 현재 자신의 상태를 저장할 수 있으며, 테이프 한 칸의 값을 쓰고 읽을 수 있음.
- 현재 자신의 상태와 테이프가 가진 값을 이용하여 좌우로 움직임.
- 지금 모든 컴퓨터는 이 방식을 따름 - 코드, 코드 포인터, 메모리, 저장 장치 등을 이용하여 여러 포인터를 조작하고 값을 출력함.
- 튜링 완전 : 튜링 기계가 할 수 있는 모든 짓을 다 할 수 있는 것을 뜻함.

  예 : C언어는 튜링 완전한 언어이다 = C언어로 튜링 기계가 할 수 있는 짓을 다 할 수 있다.
- 토막상식 : 믿기지 않겠지만 HTML+CSS는 튜링 완전합니다.

Brainfuck?

- 최소한의 문법을 이용해 프로그래밍하는 프로그래밍 언어.
- Urban Müller가 1993년에 제작.
- 튜링 완전한 언어임.

  모든 튜링 기계를 만들 수 있음 → 컴퓨터가 할 수 있는 일은 다 할 수 있음

  (실제로 Basic to Brainfuck converter, C to Brainfuck converter 등이 존재함.)

코드 예시

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

- 위 코드를 Brainfuck 실행기에 넣으면 "Sparcs!"라고 출력함.
- 저 코드가 어떻게 작동하는 건지는 이제 배울 것임.
- 물론 문자열을 출력하는 간단한 것 말고도, 배열을 정렬하거나, 잉여력이 넘쳐난다면 게임을 만들 수도 있음.

Brainfuck (인터프리터)의 구조

- Brainfuck에는 단 8개의 문자만이 쓰임. (나머지는 무시)
- Brainfuck은 정수를 담는 하나의 배열과, 그 배열의 칸 하나를 가리키는 포인터로 이루어져 있음.

- 배열의 숫자들은 모두 0으로 초기화되어 있음.
- 명령어들은 포인터를 좌우로 움직이거나, 포인터가 가리키는 숫자를 변경하는 등의 작업을 함.

Brainfuck 문법

- `+`, `-` : 포인터가 가리키는 숫자를 1 증가시키거나 감소함.
- `>`, `<` : 포인터를 오른쪽 또는 왼쪽으로 움직임.
- `.` : 포인터가 가리키는 값을 문자로 출력함.
- `,` : 문자 하나를 입력받아 포인터가 가리키는 곳에 저장.
- `[` : while(포인터가 가리키는 값 ≠ 0){
- `]` : }
- 이게 전부임. ㅇㅇ.

예제 1

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

- 처음 칸을 '칸 0', 다음 칸을 '칸 1'이라고 하면...
- `++++++++` : 칸 0에 8을 집어넣음.
- `[-]` : 현재 칸이 0이 아니라면...
- `>++++++++` : 오른쪽으로 가서 8을 증가시킨 다음...
- `<-` : 왼쪽으로 가서 1을 감소시켜라.
- 결과적으로 `[>++++++++<-]`는 오른쪽 칸에 현재 칸에 적힌 수의 8배를 채워넣고, 현재 칸을 0으로 만듦.
- `++++++++[>++++++++<-]>+` : 칸 1에 8×8+1 = 65(문자 'A')를 채워넣음.

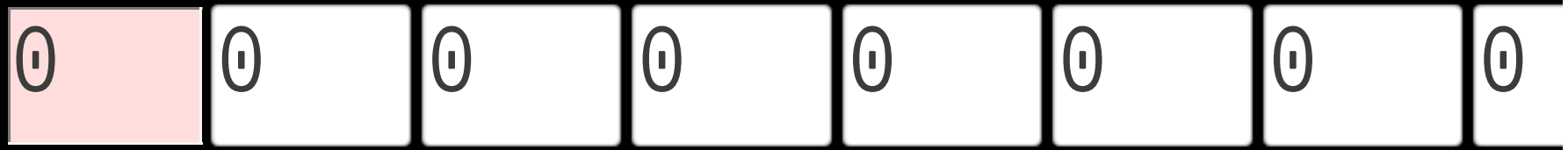
예제 1.5 : AAAAAAAAAA

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

- `[.]` : 현재 칸에 있는 숫자가 0이 아니라면 그것을 문자로 출력함. 그리고 그걸 반복함.
- 이 프로그램은 절대로 끝나지 않고, 'A'를 무한히 출력함.

기본 테크닉

- 곱하기는 예제 1에서 봤듯이 두 개의 칸을 활용함. (다만, 원래 칸에 적힌 수는 없어짐.)
- `[>+<-]`로 다른 칸으로 이동이 가능함. (`>`와 `<`의 수를 적당히 조절하여 멀리 있는 칸으로 값을 옮길 수 있음.)
- 이것을 응용하여 여러 칸에 숫자를 분배하거나 (예: 5,41,24를 만드려면 `+++++[>+>++++++++>+++++<<<]>>+>-`) 두 칸의 숫자를 합치는 것도 가능함.
- 문자열 출력은 곱셈과 숫자 분배를 적절히 사용하여 짧게 할 수 있음.
- ex. '*'(42)을 출력하려면 `++++++++++ ++++++++++ ++++++++++ ++++++++++ ++.`같은 노가다 보다 `+++++++[>++++++<-]>.`가 짧고 더 좋음.



예제 2 : Sparcs!

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

- 빨간 부분 : 전체적인 초기화, 셀 1을 8×10=80, 2를 8×14=112로 초기화 하고 있음.
- 그 다음 셀 1에 3을 더해 83('S')을 만든뒤 출력하고, 셀 2에 있는 값인 112('p')를 출력함.
- 셀 1에 14를 더해 97('a')을 만들어 출력시키고, 그 뒤에 차례대로 112+2('r'), 97+2('c'), 114+1('s')를 출력함.
- 초록색 부분에서는 셀 1에 있는 값인 99를 3으로 나눈 값(33,'!')을 셀 0에 넣은 다음, 셀 0을 출력시키고 있음.
- 결과 : **Sparcs!**

실습 : 9부터 0까지 출력하기

- http://www.lordalcol.com/brainfuckjs/
- '9'(57)부터 '0'(48)까지 순서대로 출력하는 프로그램을 짠다.
- 1. 반복문을 쓰지 않고 하나씩 출력시켜보기
- 2. 반복문을 사용하여 출력시켜보기
- 힌트 : 48을 빼고 비교한 다음 더하기.
- 일단 '9'는 `++++++[>++++++++++<-]>---.`
- 48을 빼본 다음 0이 아니면 다시 48을 더하는 식으로 어떤 수가 48인지 아닌지 알 수 있다!
- 48 빼기 : `<++++++++[>------<-]>`, 다시 더하기 : `<++++++++[>++++++<-]>`

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

실습 : 문자 출력하기

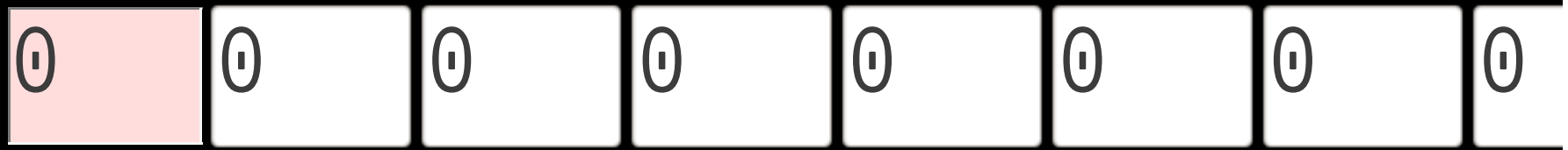
- 문자를 `,`로 입력받아, 그 문자로 이루이전 직각삼각형 출력하기.
- 첫번째 줄에는 그 문자의 코드값만큼 문자가 존재하며, 줄 아래로 갈 수록 문자의 수가 하나씩 줄어듦.

예제 3 : 별

- 아래 프로그램은 별을 1개, 2개, 3개, ..., 99개, 100개, 99개, ..., 2개, 1개 포함한 줄을 출력한다.

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

- 이렇게 많이 복잡해 보이는 코드는 그림을 그려넣고 분석하면 된다.



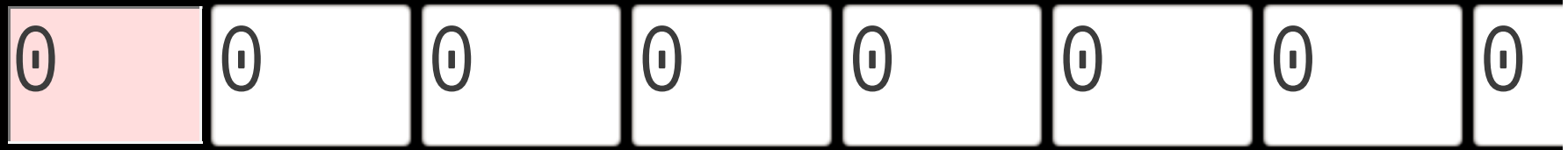
- 첫번째 줄 : 초기화
- 두세번째 줄 : 별을 1~99개 출력
- 네번째 줄 : 별을 100~1개 출력

if-else문

- 0번째 칸에 포인터가 있고, 1번째, 2번째 칸이 비어있다면
- if(x){코드1}else{코드2} : `[>+>+<<-]>[<+>-]+ >[코드1<->[-]]<[코드2[-]]`
- 또 다른 방법 : `>+<[코드1>-]>[<코드2>-]<<`

나눗셈

- 0번째 칸에 나눠질 수, 1번째 칸에 나눌 수가 있고 0번째 칸에 포인터가 있다면
- `[->-[>+>>]>[+[-<+>]>+>>]<<<<<]` 를 실행하면 2번째 칸, 3번째 칸에 나머지와 몫이 저장된다.



- 0번째 칸과 1번째 칸에서 동시에 수를 하나씩 빼 가며 1번째 칸이 0이 되었을 때 몫을 1 늘인다.

참고하면 매우 좋을 자료

- Wikipedia : Brainfuck (http://en.wikipedia.org/wiki/Brainfuck)
- Esolangs : Brainfuck algorithms (http://esolangs.org/wiki/Brainfuck_algorithms)
- Brainfuck Interpreter (http://www.lordalcol.com/brainfuckjs/)
- Lost Kingdom (http://jonripley.com/i-fiction/games/LostKingdomBF.html)

끝

- 당신은 이제 Ook!, COW, Pi, PATH... 등의 프로그래밍 언어를 쉽게 구사할 수 있을 겁니다.
- 더 많은 난해한 프로그래밍 언어들 : esolangs.org에서 찾아볼 수 있음.
- 아희, Fractran, Befunge, Piet, Whitespace, Chef, LOLCODE, **Malbolge** ....

몇 가지 소스 코드들

- Brainfuck으로 구현한 QuickSort
- http://codegolf.stackexchange.com/questions/2445/implement-quicksort-in-brainf

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

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

- Quine
- http://www.hevanet.com/cristofd/brainfuck/

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

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