Лекция 3: Brainfuck (часть 1)

«Технология разработки программного обеспечения»

Язык Brainfuck

Для понимания языка и удобства программирования лучше всего использовать визуализатор https://fatiherikli.github.io/brainfuck-visualizer

Brainfuck это эзотерический язык программирования; он разработан для исследования границ возможностей языков программирования, для доказательства утверждения, для искусства, для юмора.

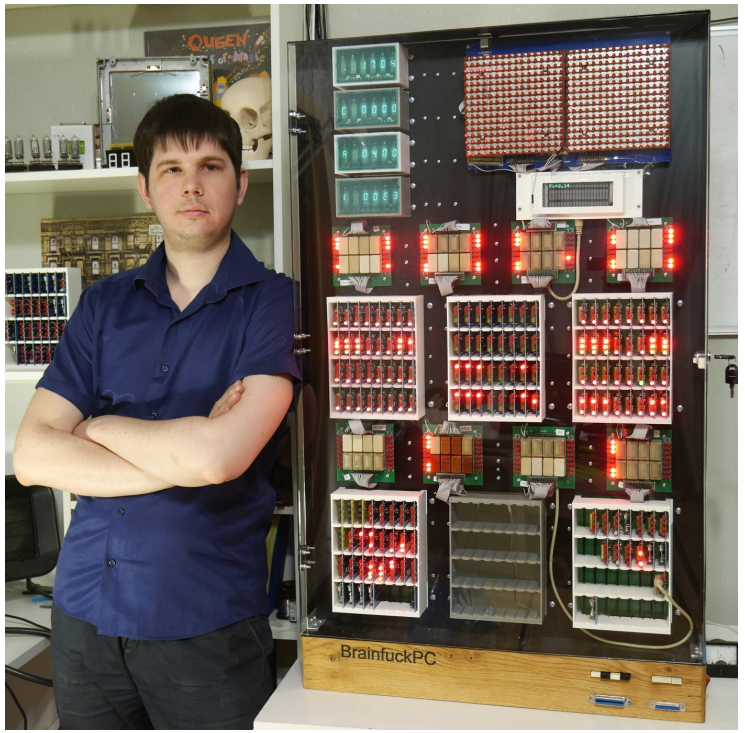
Язык Brainfuck

Автор Урбан Мюллер, 1993 год. Язык имеет 8 команд, каждая записывается 1 символом. Исходный код программы это последовательность символов без синтаксиса. Компилятор Brainfuck занимает примерно 200 байт.

Подробно https://ru.wikipedia.org/wiki/Brainfuck

Язык Brainfuck

Brainfuck управляет "машиной", которая состоит из упорядоченного набора ячеек и указателя на текущую ячейку. Язык имеет тьюринговую полноту, т.е. **Brainfuck полноценный язык программирования.**



Сумрачный гений

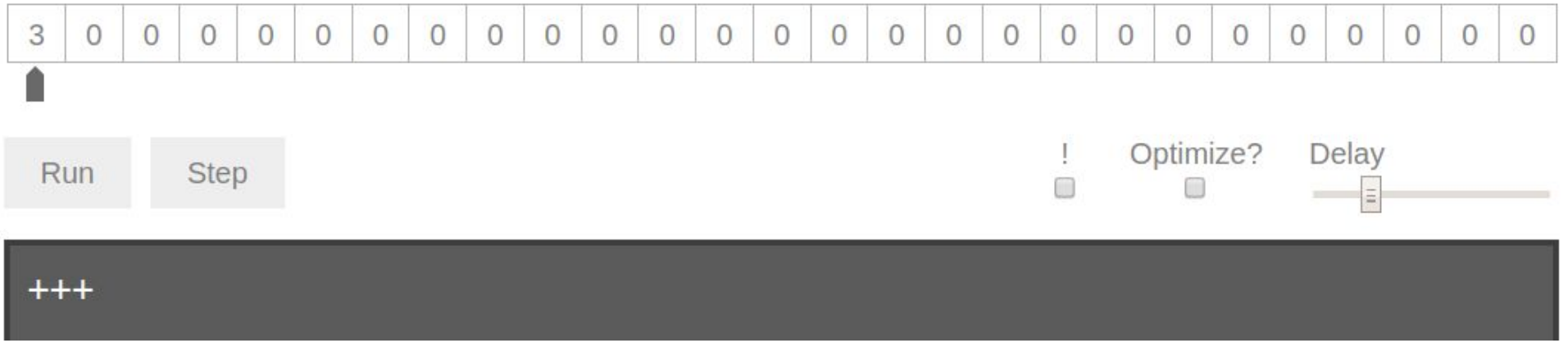
Правила языка

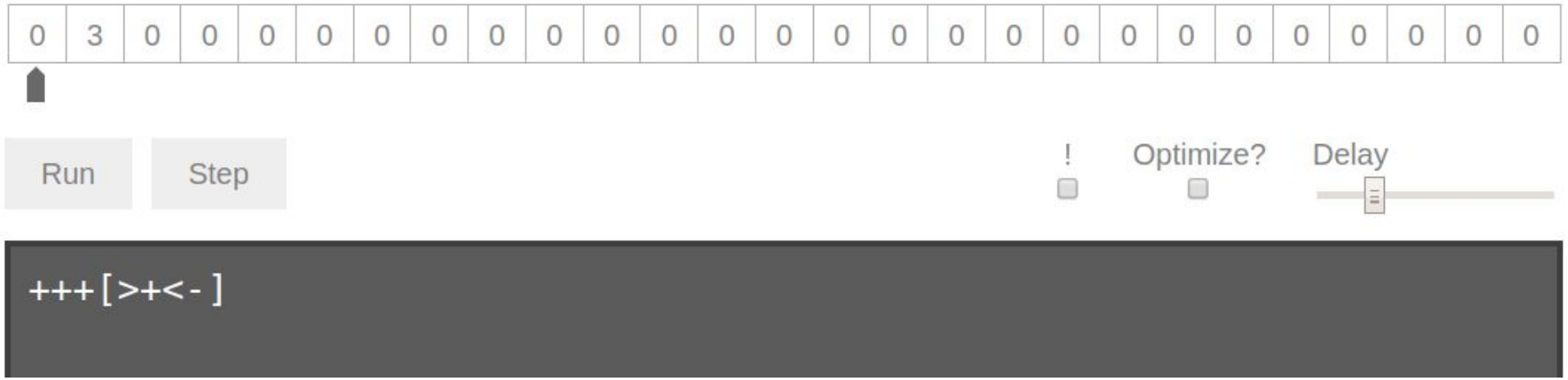
1) Одна ячейка = 1 байт
2) На старте 30,000 ячеек
3) Старт это крайняя левая позиция
4) Ввод и вывод идет ASCII-кодом
5) Число 1 будет записано как 0x31 (49)
6) Если в ячейке находится 0x41 (65), то на экран выведется символ «A» (большая английская буква A)

Команды языка

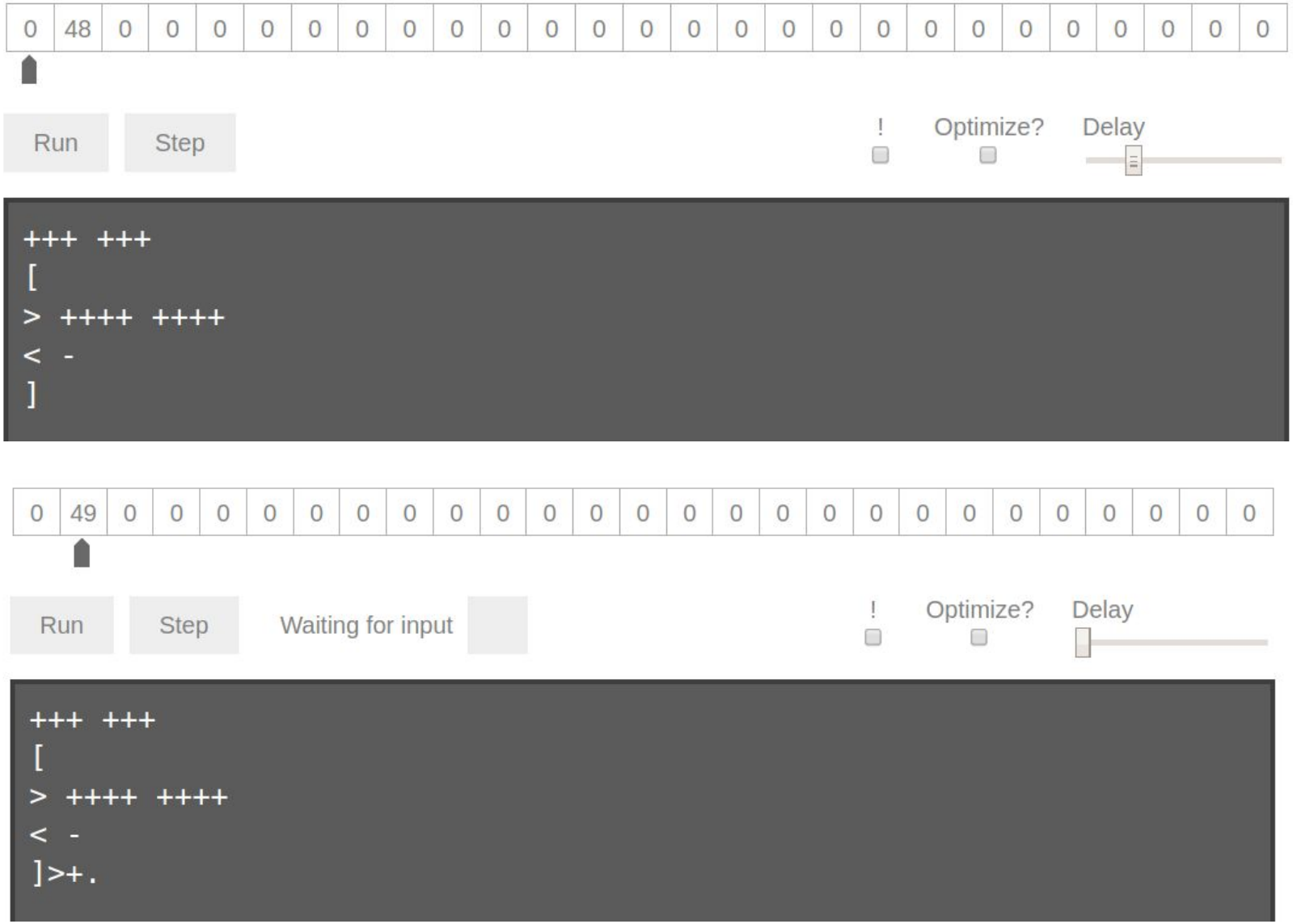
```
>     перейти к следующей ячейке
<     перейти к предыдущей ячейке
+     увеличить значение в ячейке на 1
-     уменьшить значение в ячейке на 1
.     напечатать значение из текущей ячейки
,     ввести значение и сохранить его в ячейке
[     цикл: если текущее значение ячейки = 0, то
            перейти на ] (нет захода в цикл)
]     цикл: если текущее значение ячейки !=0, то
            перейти назад на [ (на начало цикла)
```

Примеры


3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Run
Step
!
Optimize?
Delay
+++

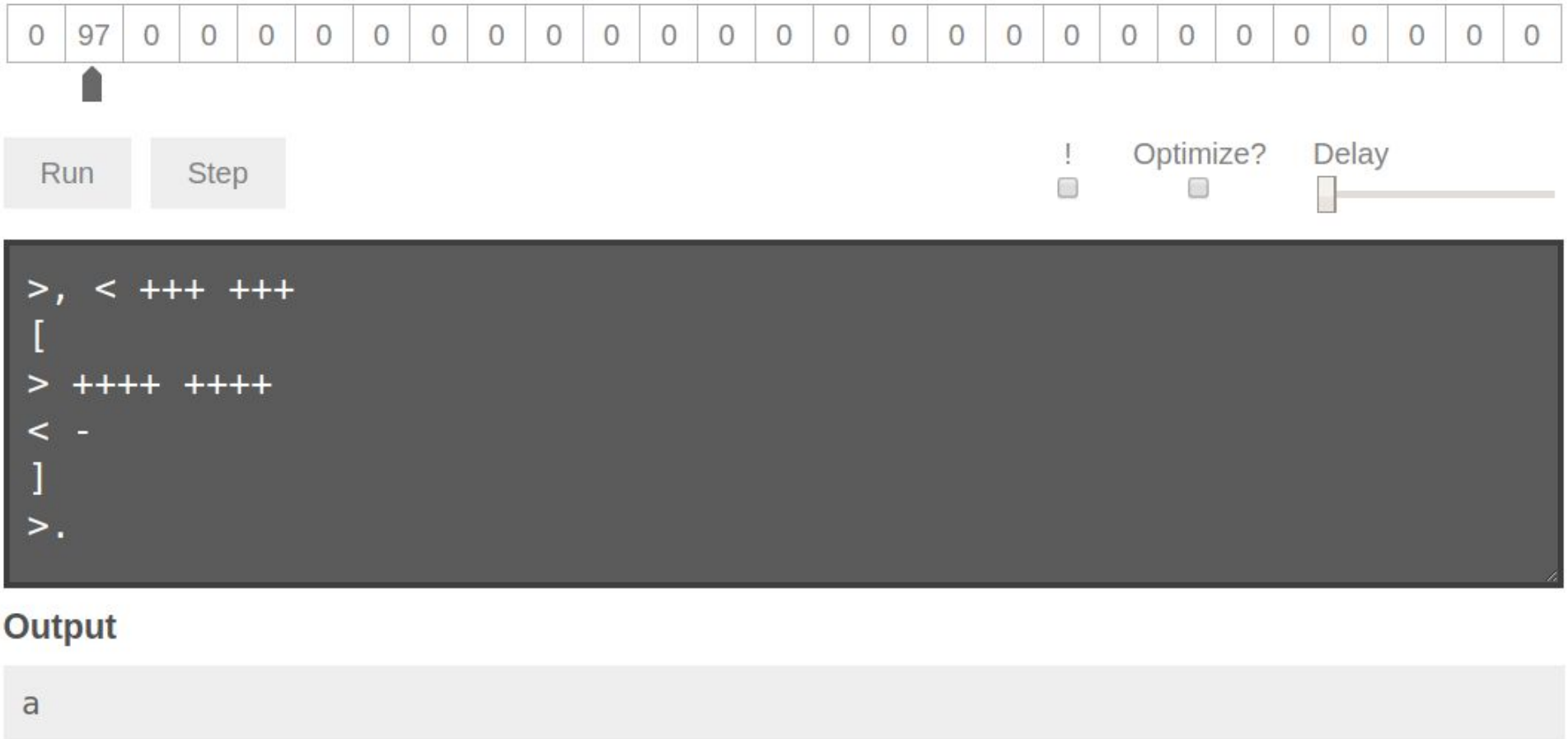

0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Run
Step
!
Optimize?
Delay
+++[>+<-]

Примеры


0 48 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Run
Step
!
Optimize?
Delay
+++ +++
[
> ++++ ++++
< -
]
0 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Run
Step
Waiting for input
!
Optimize?
Delay
+++ +++
[
> ++++ ++++
< -
]>+.

Примеры

Как вывести на экран английскую букву «a»?



Hello World!

Как вывести на экран Hello World!

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

Как это все работает? Что обозначают эти плюсики, точки и скобочки?

Hello World!

Мы хотим вывести строку

```
63
64 H   e    l    l    o    _    W    o    r    l    d    !
65 72  101  108  108  111  32   87   111  114  108  100  33
```

Для этого в первые ячейки ленты занесем некоторые вспомогательные числа

```
67
68          70   100  30   10
69 [0]  [1]  [2]  [3]  [4]  [5]
```

Зачем нам эти числа?

Hello World!

Подготовим первые 4 ячейки, занесем туда числа 70, 100, 30, 10

```
++++++++++  присваивание ячейке 0 значение 10
[           повторять, пока значение текущей ячейки > нуля
>+++++++    приращение ячейки 1 на 7
>++++++++++ приращение ячейки 2 на 10
>+++        приращение ячейки 3 на 3
>+          приращение ячейки 4 на 1
<<<<-       возврат к ячейке 0 и его уменьшение на 1
]           вернуться к началу цикла
```

Hello World!

А теперь выводим символы

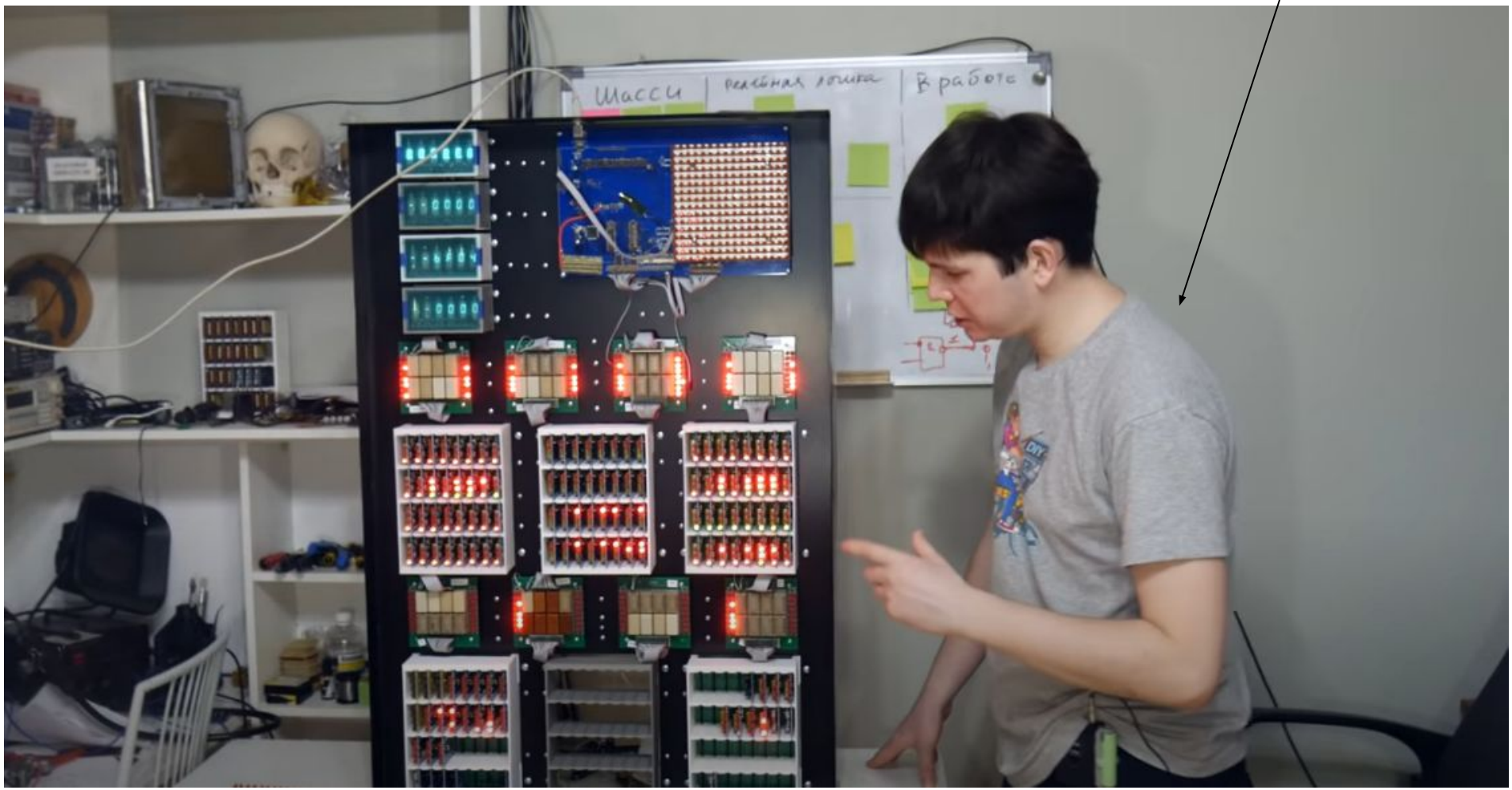
```
>++.            Вывод «H». Получение кода «H» (72)
>+.             Вывод «e». Получение кода «e» (101)
+++++++..       Вывод «ll». Получение кода «l» (108)
+++.            Вывод «o». Получение кода «o» (111)
>++.            Вывод пробела. Получение кода пробела (32)
<<++++++++++++++.   Вывод «W». Получение кода «W» (87)
>.              Вывод «o». Код «o» (111)
+++.            Вывод «r». Получение кода «r» (114)
------.         Вывод «l». Получение кода «l» (108)
--------.       Вывод «d». Получение кода «d» (100)
>+.             Вывод «!». Получение кода «!» (33)
>.              Вывод кода перевода строки (10)
```

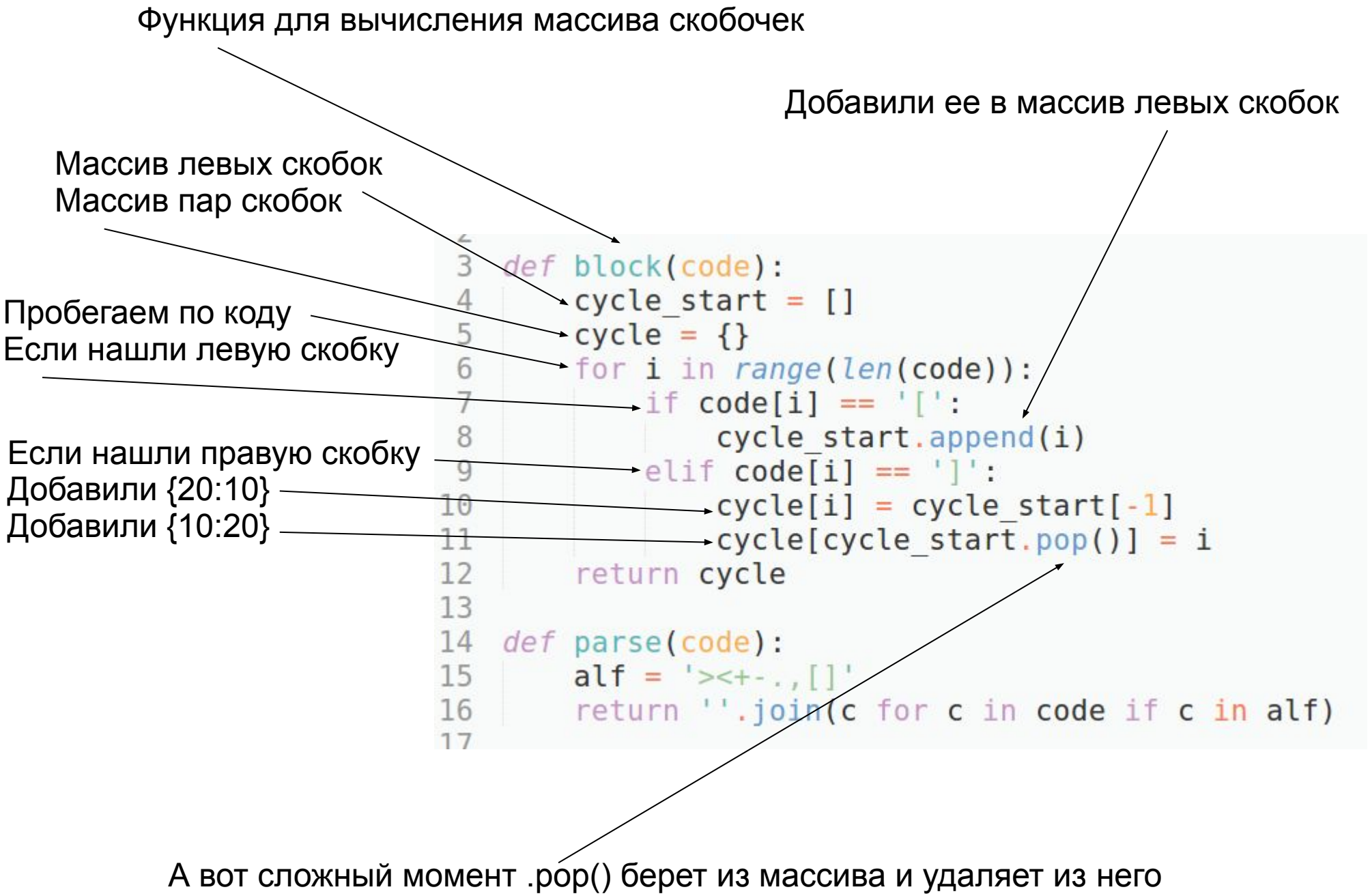
```
+++++ +++++               initialize counter (cell #0) to 10
[                         use loop to set the next four cells to 70/100/30/10
    > +++++ ++                add  7 to cell #1
    > +++++ +++++             add 10 to cell #2
    > +++                     add  3 to cell #3
    > +                       add  1 to cell #4
    <<<< -                    decrement counter (cell #0)
]
> ++ .                    print 'H'
> + .                     print 'e'
+++++ ++ .                print 'l'
.                         print 'l'
+++ .                     print 'o'
> ++ .                    print ' '
<< +++++ +++++ +++++ .    print 'W'
> .                       print 'o'
+++ .                     print 'r'
----- - .                 print 'l'
----- --- .               print 'd'
> + .                     print '!'
> .                       print '\n'
```

Релейный компьютер

Тоже сумрачный гений

https://clck.ru/RrkWk




Функция для вычисления массива скобочек
Добавили ее в массив левых скобок
Массив левых скобок
Массив пар скобок
Пробегаем по коду
Если нашли левую скобку
Если нашли правую скобку
Добавили {20:10}
Добавили {10:20}
def block(code):
cycle_start = []
cycle = {}
for i in range(len(code)):
if code[i] == '[':
cycle_start.append(i)
elif code[i] == ']':
cycle[i] = cycle_start[-1]
cycle[cycle_start.pop()] = i
return cycle
def parse(code):
alf = '><+-.,[]'
return ''.join(c for c in code if c in alf)
А вот сложный момент .pop() берет из массива и удаляет из него

Самый сложный момент

```
219
220 + + [ > + [ - ] - ]  .
221 0 1 2 3 4 5 6 7 8 9 10
222
223     cycle = {}
224     cycle_start = []
225
226 0   ничего
227 1   ничего
228 2   cycle_start.append(2) = [2]
229 3   ничего
230 4   ничего
231 5   cycle_start.append(5) = [2,5]
232 6   ничего
233 7   cycle добавили {7:5}
234     cycle добавили {5:7} и удалили 5-ку
235     cycle_start = [2]
236 8   ничего
237 9   аналогично что в п.7
238     cycle добавили {9:2}
239     cycle добавили {2:9} и удалили 2-ку
240 10  ничего
241 вернули из функции
242
```

```
def block(code):
    cycle_start = []
    cycle = {}
    for i in range(len(code)):
        if code[i] == '[':
            cycle_start.append(i)
        elif code[i] == ']':
            cycle[i] = cycle_start[-1]
            cycle[cycle_start.pop()] = i
    return cycle
```

Вот что пришло
Вот что должно уйти

```
{7: 5, 5: 7, 9: 2, 2: 9}
```

```
21 def run(code):
22     code = parse(code)
23     x = 0
24     i = 0
25     codeline = {0: 0}
26     cycle = block(code)
27     while i<len(code):
28         litera = code[i]
29         if litera == '>':
30             x += 1
31             codeline.setdefault(x,0)
32         elif litera == '<':
33             x -= 1
34         elif litera == '+':
35             codeline[x] += 1
36         elif litera == '-':
37             codeline[x] -= 1
38         elif litera == '.':
39             print(chr(codeline[x]), end='')
40         elif litera == ',':
41             codeline[x] = int(input('0-255: '))
42         elif litera == '[':
43             if codeline[x] == 0: i = cycle[i]
44         elif litera == ']':
45             if codeline[x] !=0: i = cycle[i]
46         i += 1
47
48 code = input()
49 run(code)
50
```

Код = парсим(код)

Текущая ячейка

Цикловая ячейка

Массив {ячейка:значение}

Делаем массив скобок

Если >, то идем вправо
Если там нет значения,
то устанавливаем его = 0

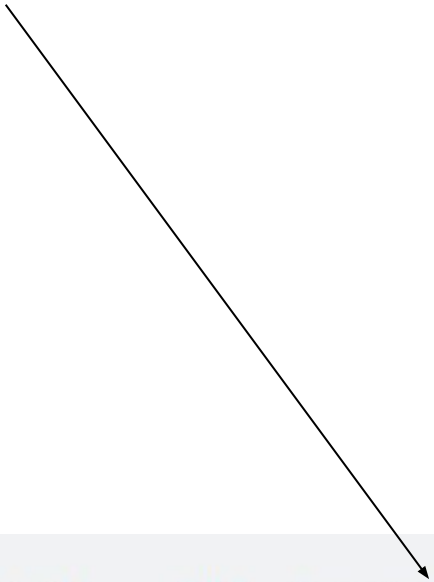
Если ., то выводим код символа,
который там (65 → A)

Если , то просим ввести
Число 0-255 (хотя, конечно,
надо вводить символ)

Прыжок

Прыжок

Код Hello World!



```
214
215 code = """ ++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++
216 .>+.+++++++..+++.>++.<<+++++++++++++++.>.+++
217 .------.--------.>+.>. """
218
219 run(code)
220
```

Задание по лекции 3

1) Набрать компилятор, проверить Hello World!
2) Создать программу для вывода своего имени (русскими буквами с заглавной буквы)
3) Проверить код онлайн и через свой компилятор