

## Семинар № 8. «Делегаты»

### 1. Практика «Виртуальная машина Brainfuck»

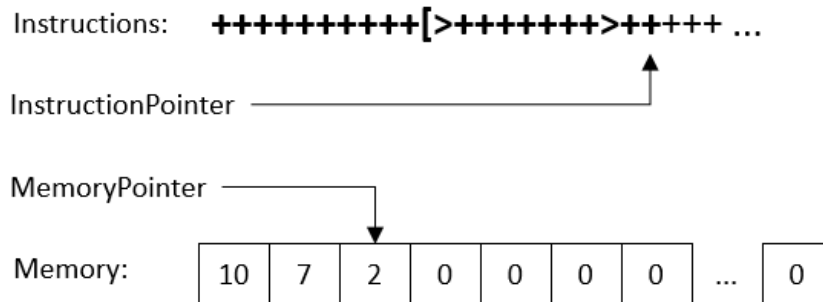
Скачайте проект **brainfuck**.

Создавать языки программирования сложно. Но не всегда! Язык программирования **Brainfuck** — это экстремально простой язык программирования, интерпретатор при желании можно уместить на один экран кода. Программа на **Brainfuck** состоит только из символов `+-<>.,[]`, поэтому читать такие программы не очень удобно. :-)

В этой серии задач вам предстоит создать этот интерпретатор с возможностью его простого расширения новыми операциями.

В задачах программирования интерпретаторов часто оказываются удобными пройденные в этом блоке делегаты анонимные функции — решите эту задачу с помощью анонимных функций.

#### Виртуальная машина



Виртуальная машина хранит следующее:

- Массив памяти, каждая ячейка которого хранит **1 байт**. По умолчанию размер памяти — **30000** ячеек.
- Указатель на текущую ячейку памяти. Изначально, указатель указывает на нулевую ячейку.
- Выполняемую программу. Она состоит из инструкций, каждая обозначается одним символом. Программа начинает выполняться с первого символа последовательно.
- Номер выполняемой в данный момент инструкции. После выполнения любой инструкции номер увеличивается на единицу. Как только номер инструкции выходит за пределы программы, выполнение заканчивается.
- Конкретные операции на языке **Brainfuck** могут читать или менять эти данные.

В этой части вам нужно реализовать виртуальную машину в классе **VirtualMachine.cs** так, чтобы проходили все тесты из файла **VirtualMachineTests.cs**.

## 2. Практика «Виртуальная машина Brainfuck»

Продолжайте работу в том же проекте [brainfuck](#).

Изучите класс **Brainfuck.cs**, в частности то, как он использует реализованный ранее класс **VirtualMachine**.

В классе **BrainfuckBasicCommands** реализуйте метод, регистрирующий следующие простые команды в виртуальную машину:

Символ	Значение
.	Вывести байт памяти, на который указывает указатель, преобразовав в символ согласно ASCII
+	Увеличить байт памяти, на который указывает указатель
-	Уменьшить байт памяти, на который указывает указатель
,	Ввести символ и сохранить его ASCII-код в байт памяти, на который указывает указатель
>	Сдвинуть указатель памяти вправо на 1 байт
<	Сдвинуть указатель памяти влево на 1 байт
A-Z, a-z, 0-9	сохранить ASCII-код этого символа в байт памяти, на который указывает указатель

Например, программа `++>+++.<` выводит два символа с ASCII кодами **2** и **3**, а память после выполнения команды будет выглядеть так `[2, 3, 0, 0, ... 0]`.

Для ввода и вывода используйте переданные в метод **Run** функции **Func<int> read** и **Action<char> write**.

Тут **read** по аналогии с **Console.Read** возвращает либо код введенного символа, либо -1, если ввод закончился. Считайте, что на вход будут подаваться только символы с кодами **0..255** — они точно помещаются в один байт.

Детали реализации инструкций восстановите по тестам в классе **BrainfuckBasicCommandsTests**. Сделайте так, чтобы все тесты в этом файле проходили.

### 3. Практика «Циклы Brainfuck»

Продолжайте работу в том же проекте [brainfuck](#).

В классе **BrainfuckLoopCommands** реализуйте метод, регистрирующий следующие команды в виртуальную машину:

Символ	Значение
[	(Начало цикла) Перескочить по программе вправо на соответствующий (с учетом вложенности) символ ], если текущий байт памяти равен нулю. Продолжать исполнение с этого символа.
]	(Конец цикла) Перескочить по списку инструкций влево на соответствующий (с учетом вложенности) символ [, если текущий байт памяти НЕ равен нулю. Продолжать исполнение с этого символа.

Например, программа `+++++++[>+++++++<-]>+.` выводит букву **A** (ASCII-код 65 получается увеличением 8 раз второй ячейки на 8, а потом добавлением ещё единицы).

Детали реализации инструкций восстановите по тестам в классе **BrainfuckLoopCommandsTests**. Сделайте так, чтобы все тесты в этом файле проходили.