

Brainfuck

Материал из Википедии — свободной энциклопедии

Brainfuck (англ. *brain* мозг + *fuck*) — один из известнейших эзотерических языков программирования , придуман Урбаном Мюллером (нем. *Urban Müller*) в 1993 году для забавы. Язык имеет восемь команд, каждая из которых записывается одним символом. Исходный код программы на *Brainfuck* представляет собой последовательность этих символов без какого-либо дополнительного синтаксиса.

Одним из мотивов Урбана Мюллера было создание языка с как можно меньшим компилятором. Отчасти он был вдохновлен языком FALSE, для которого существовал компилятор размера 1024 байта. Существуют компиляторы языка Brainfuck размера меньше 200 байт. Программы на языке Brainfuck писать сложно, за что его иногда называют языком для мазохистов. Но при этом важно отметить, что Brainfuck является вполне естественным, полным и простым языком и может использоваться при определении понятия вычислимости.

Машина, которой управляют команды *Brainfuck*, состоит из упорядоченного набора ячеек и указателя текущей ячейки, напоминая ленту и головку машины Тьюринга . Кроме того, подразумевается устройство общения с внешним миром (см. команды . и ,) через поток ввода и поток вывода.

Язык Brainfuck можно описать с помощью эквивалентов языка Си (предполагается, что переменная p объявлена как указатель на байт):

Команда Brainfuck	Эквивалент на Си	Описание команды
>	++p;	перейти к следующей ячейке
<	--p;	перейти к предыдущей ячейке
+	++(*p);	увеличить значение в текущей ячейке на 1
-	--(*p);	уменьшить значение в текущей ячейке на 1
.	putchar(*p);	напечатать значение из текущей ячейки
,	*p = getchar();	ввести извне значение и сохранить в текущей ячейке
[if (*p) do {	если значение текущей ячейки нуль, перейти вперёд по тексту программы на ячейку, следующую за соответствующей] (с учётом вложенности)
]	} while(*p);	переход назад по тексту программы на символ [(с учётом вложенности)

Несмотря на внешнюю примитивность, *Brainfuck* с бесконечным набором ячеек имеет тьюринговскую полноту , а, следовательно, по потенциальным возможностям не уступает «настоящим» языкам, подобным Си, Паскалю или Java.

Brainfuck подходит для экспериментов по генетическому программированию из-за простоты синтаксиса, и, соответственно, генерации исходного кода.

В «классическом» *Brainfuck*, описанном Мюллером, размер ячейки — один байт, количество ячеек 30 000. В начальном состоянии указатель находится в крайней левой позиции, а все ячейки заполнены нулями. Увеличение/уменьшение значений ячеек происходит по модулю 256. Ввод/вывод также происходит побайтно, с учётом кодировки ASCII (то есть в результате операции ввода (,) символ **1** будет записан в текущую ячейку как число 0x31 (49), а операция вывода (.), совершённая над ячейкой, содержащей 0x41 (65), напечатает латинскую **A**). В других вариантах языка размер и количество ячеек может быть другим (большим). Есть версии, где значение ячеек не целочисленно (с плавающей точкой).

Содержание

- 1 Пример программы
- 2 Программирование на языке Brainfuck
- 3 Языки на основе Brainfuck
- 4 См. также
- 5 Дialectы и реализации
 - 5.1 Другие абстрактные исполнители и формальные системы вычислений
- 6 Ссылки

Пример программы

Программа на языке Brainfuck, печатающая «*Hello World!*»:

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

Разбор программы:

Подготовка в памяти (с ячейки 1) массива значений, близких к ASCII-кодам символов, которые необходимо вывести (70, 100, 30, 10), через повторение 10 раз приращения ячеек на 7, 10, 3 и 1, соответственно

++++++	присваивание ячейке 0 (счетчику) значения 10
[повторять, пока значение текущей ячейки (ячейки 0) больше 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 (счетчику), и его уменьшение на 1
]	вернуться к началу цикла

Получение кодов букв и их вывод

>+.	Вывод «H». Получение кода «H» (72) из 70 в ячейке 1 и вывод
>+.	Вывод «e». Получение кода «e» (101) из 100 в ячейке 2 и вывод
+++++..	Вывод «ll». Получение кода «l» (108) из 101 в ячейке 2 и вывод дважды
+++.	Вывод «o». Получение кода «o» (111) из 108 в ячейке 2 и вывод
>+.	Вывод пробела. Получение кода пробела (32) из 30 в ячейке 3 и вывод
<+++++++.	Вывод «W». Получение кода «W» (87) из 72 в ячейке 1 и вывод
>.	Вывод «o». Код «o» (111) уже находится в ячейке 2, просто его выводим
+++.	Вывод «r». Получение кода «r» (114) из 111 в ячейке 2 и вывод
-----.	Вывод «l». Получение кода «l» (108) из 114 в ячейке 2 и вывод
-----.	Вывод «d». Получение кода «d» (100) из 108 в ячейке 2 и вывод
>+.	Вывод «!». Получение кода «!» (33) из 32 в ячейке 3 и вывод
>.	Вывод кода перевода строки (10) из ячейки 4

В принципе, печать «Hello World!» можно реализовать проще, но программа будет в три с лишним раза больше, чем приведённый выше оптимизированный вариант:

```
+++++
+++++ .+++++
+++++ .+++++ .+++ .-----
-----
----- .+++++
+++++ .+++++
+++++ .+++ .-----
-----
----- .
```

Программирование на языке Brainfuck

Каждый начинающий программировать на *Brainfuck* немедленно сталкивается со следующими проблемами:

- отсутствие операции копирования значения
- отсутствие промежуточной (*аккумуляторной*) памяти
- отсутствие условных операторов в их привычном виде
- отсутствие привычной арифметики, операций умножения и деления

Эти проблемы могут быть решены.

```
обозначим за @(k) сдвиг на k ячеек вправо, если k>0, и влево, если k<0
Соответственно, @(k) = >...k раз...> либо <...-k раз...<
zero(): обнуление текущей ячейки:
[ - ]
=
[ + ]
add(k): прибавление значения ячейки n (текущей) к значению ячейки n+k:
[ - @(k) + @(-k) ]
при этом значение ячейки n теряется (обнуляется).
mov(k): копирование значения ячейки n (текущей) в ячейку n+k с потерей (обнулением) значения ячейки n:
@(k) zero() @(-k) add(k)
=
@(k) [ - ] @(-k) [ - @(k) + @(-k) ]
copy(k, t): копирование значения ячейки n (текущей) в ячейку n+k
с использованием промежуточной ячейки n+k+t, благодаря чему значение ячейки n не теряется (сохраняется).
@(k) zero() @(t) zero() @(-k-t) [ - @(k) + @(t) + @(-k-t) ] @(k+t) mov(-k-t)
=
@(k) [ - ] @(t) [ - ] @(-k-t) [ - @(k) + @(t) + @(-k-t) ] @(k+t) [ - @(-k-t) + @(k+t) ]
ifelse(t): если текущая ячейка>0, то выполняется условие true
            если текущая ячейка=0, то выполняется условие false
            t-относительный номер вспомогательной ячейки:
@(t)[-]+@(-t) устанавливаем флаг 1 для случая else
[
    здесь действия ветки true
    @(t)[-]@(-t) устанавливаем флаг 0 для случая else
    [-] выход из цикла
]
@(t)
[ @(-t)
    здесь действия ветки false
    @(t)[-] выход из цикла
```

Brainfuck почти не используется для практического программирования (за исключением работ отдельных энтузиастов), а используется преимущественно для головоломок и задач для соревнований.

Языки на основе Brainfuck

Ook!	COW	Brainfuck	Описание
Ook. Ook.	MoO	+	Значение текущей ячейки увеличивает на 1
Ook! Ook!	MOo	-	Значение текущей ячейки уменьшают на 1
Ook. Ook?	moO	>	Следующая ячейка
Ook? Ook.	mOo	<	Предыдущая ячейка
Ook! Ook?	moo	[Начало цикла
Ook? Ook!	MOO]	Конец цикла
Ook! Ook.	OOM	.	Вывод значения текущей ячейки
Ook. Ook!	oom	,	Запрос значения текущей ячейки
	mOO	<i>отсутствует</i>	Выполняет инструкцию с номером, который берётся из текущей ячейки (если значение в текущей ячейке равно 3, то это приводит к бесконечному циклу).
	MOo	<i>отсутствует</i>	Если значение текущей ячейки равно нулю, то ввести его с клавиатуры; если же значение текущей ячейки — не ноль, то вывести его на экран.
	OOO	<i>отсутствует</i>	Обнуляет значение в текущей ячейке
	MMM	<i>отсутствует</i>	

См. также

- Тьюринговская трясина

Диалекты и реализации


- BrainSub (<http://progopedia.ru/dialect/brainsub>)
- Brainfork (<http://ru.wikipedia.org/wiki/Brainfork>)
- Brainloller (<http://progopedia.ru/dialect/brainloller>)
- COW (<http://progopedia.ru/dialect/cow>)
- Ook! (<http://progopedia.ru/dialect/ook>)
- Pbrain (<http://progopedia.ru/dialect/pbrain>)
- Smallfuck (<http://progopedia.ru/dialect/smallfuck>)
- Spoon
- LOLCODE
- Whitespace
- DoubleFuck
- Feckfeck

Другие абстрактные исполнители и формальные системы вычислений

- Алгоритм Маркова (продукционное программирование)
- Машина Тьюринга (автоматное программирование)
- Машина Поста (автоматное программирование)
- Рекурсивная функция (теория вычислимости)
- Лямбда-исчисление (функциональное программирование)

Ссылки

- Немного о brainfuck на русском языке и пара инструментов для работы с ним (<http://bf-fan.coolpage.biz/>)
- Оригинальное описание BF на английском языке и ссылки на BF-ресурсы (<http://www.muppetlabs.com/%7Ebreadbox/bf>)

- Brainfuck interpreter with integrated debugger (IDE) for Windows (<http://www.4mhz.de/>)
-  ru_brainfucker — русское ЖЖ-сообщество любителей эзотерических языков
- статья на rsdn.ru об эзотерических языках программирования (<http://rsdn.ru/article/philosophy/languages.xml>)
- Processing_BF (<http://bolk.exler.ru/files/bf/>) — оптимизирующий интерпретатор и транслятор в PHP, написанный на языке PHP
- BrainfuckInterpreter (<http://bfinterpreter.narod.ru/>) — кроссплатформенный оптимизирующий интерпретатор на Java, переехал на Brainfuck Interpreter bfrun (<http://bfrun.berlios.de/>)
- esco — универсальный интерпретатор эзотерических языков (<http://esco.sourceforge.net/>)
- Интерпретатор brainfuck на JavaScript с открытым исходным кодом (<http://brainfuck.progopedia.ru/>)
- Интерпретатор brainfuck с открытым исходным кодом (<http://bobiczdoh.jino-net.ru/?area=download>)

Источник — «<http://ru.wikipedia.org/wiki/Brainfuck>»

- Последнее изменение этой страницы: 20:33, 8 декабря 2009.
- Текст доступен на условиях лицензии Creative Commons Attribution/Share-Alike, в отдельных случаях могут действовать дополнительные условия. Подробнее см. Условия использования.