2016软工一大作业要求

本次的作业是完成一个BF语言的简单IDE。该IDE支持源码的保存和读取，代码执行以及历史版本保留等功能。

关于BF语言的说明，请参考：<https://zh.wikipedia.org/wiki/Brainfuck>

无法科学上网的同学可以参考：<http://baike.baidu.com/view/765703.htm>

一个BF语言解释器供参考：https://copy.sh/brainfuck/

具体要求如下：

1. 本IDE使用Client-Server模式，需要有客户端与服务器端。客户端负责提供GUI界面，**服务器端负责存取与执行代码**。【强调一下，代码的保存、读取、执行等功能都是在服务器上完成的】
2. 客户端GUI界面需要包括输入输出窗口以及新建保存等必要的菜单选项。后文有对界面示意的描述。
3. 实现BF代码的执行功能。客户端将源码传到服务器端执行后，服务器端将运行结果返回客户端。**BF解析器需要自行实现**，输入是BF代码与输入数据，输出是该代码的执行结果，其中所有的输入输出都是字符串。比如一个执行两个1位数相加的程序，就有如下的输入与输出

|  |  |  |
| --- | --- | --- |
| 输入代码 | 输入数据 | 输出结果 |
| ,>++++++[<-------->-],,[<+>-],<.>. | 4 3 | 7 |

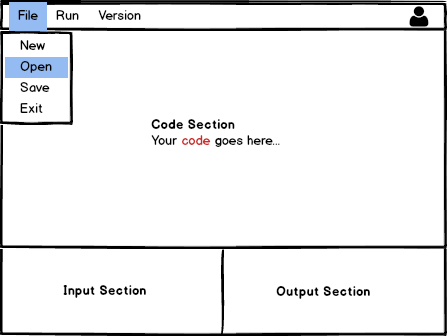
1. 实现登录登出功能以支持多用户操作。每个用户只能访问自己创建的文件。
2. 实现源码文件的历史版本保留功能，可以将代码恢复到过去某一次保存后的状态。假设用户在编辑代码时保存了三次，每保存一次就在服务器端生成一个历史版本，如A、B、C（如果两次保存之间代码没有修改则不做任何操作）。历史版本保留功能允许用户把当前代码恢复到A、B、C中任意一个版本。文件可以保留的历史版本数自定（比如只保留最后保存的十次代码）。

加分项：

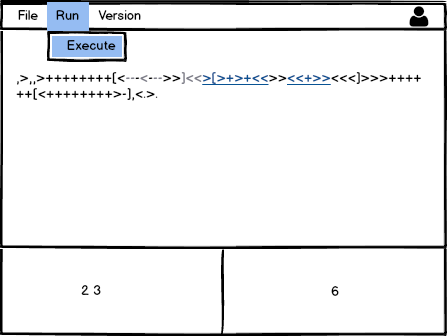
实现撤销(undo)与重做(redo)功能，不使用GUI库自带的撤销与重做功能。

界面示意：

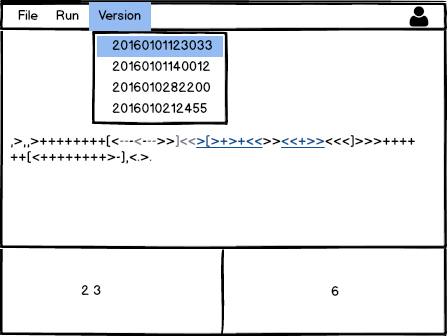
为了让同学们对整个项目有个概念，这里给出参考的示意图。**各位同学可以在满足功能的前提下自由发挥**。



界面至少要分为菜单栏，代码区，输入数据区与输出数据区。界面上有登录登出入口。



用户在输入完代码与输入数据后，点击执行。在输出数据区应当显示执行结果。这里演示的是两个一位数相乘（且结果还是一位数）的程序。



在编辑代码的过程中可以选择恢复到哪一个历史版本。具体如何在界面上实现该功能可以自定（可以如图在菜单栏中选择，也可以再添加一个区专门用来显示可用的历史版本列表等等）。

# Brainfuck[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=0" \o "编辑首段)]

维基百科，自由的百科全书

|  |  |
| --- | --- |
|  | 本条目**没有列出任何**[**参考或来源**](https://zh.wikipedia.org/wiki/Wikipedia:%E5%88%97%E6%98%8E%E6%9D%A5%E6%BA%90)。*（2015年11月13日）*  维基百科所有的内容都应该[可供查证](https://zh.wikipedia.org/wiki/Wikipedia:%E5%8F%AF%E4%BE%9B%E6%9F%A5%E8%AD%89)。 请协助添加来自[可靠来源](https://zh.wikipedia.org/wiki/Wikipedia:%E5%8F%AF%E9%9D%A0%E6%9D%A5%E6%BA%90)的引用以[改善这篇条目](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit)。[无法查证](https://zh.wikipedia.org/wiki/Wikipedia:%E5%8F%AF%E4%BE%9B%E6%9F%A5%E8%AF%81)的内容可能被提出异议而移除。 |

**Brainfuck**，是一种极小化的[计算机语言](https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%AF%AD%E8%A8%80" \o "计算机语言)，它是由[Urban Müller](https://zh.wikipedia.org/w/index.php?title=Urban_M%C3%BCller&action=edit&redlink=1)在1993年创建的。由于[fuck](https://zh.wikipedia.org/wiki/Fuck" \o "Fuck)在[英语](https://zh.wikipedia.org/wiki/%E8%8B%B1%E8%AA%9E" \o "英语)中是[脏话](https://zh.wikipedia.org/wiki/%E9%AB%92%E8%A9%B1)，这种语言有时被称为**brainf\*ck**或**brainf\*\*\***，甚至被简称为**BF**。

## 概述[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=1" \o "编辑小节：概述)]

Müller的目标是创建一种简单的、可以用最小的[编译器](https://zh.wikipedia.org/wiki/%E7%BC%96%E8%AF%91%E5%99%A8" \o "编译器)来实现的、匹配[图灵完全](https://zh.wikipedia.org/wiki/%E5%9B%BE%E7%81%B5%E5%AE%8C%E5%85%A8)思想的编程语言。这种语言由八种[运算符](https://zh.wikipedia.org/w/index.php?title=%E8%BF%90%E7%AE%97%E7%AC%A6&action=edit&redlink=1)构成，为[Amiga](https://zh.wikipedia.org/wiki/Amiga)机器编写的[编译器（第二版）](http://wuarchive.wustl.edu/pub/aminet/dev/lang/brainfuck-2.lha)只有240个[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82" \o "字节)大小。

就象它的名字所暗示的，brainfuck程序很难读懂。尽管如此，brainfuck[图灵机](https://zh.wikipedia.org/wiki/%E5%9B%BE%E7%81%B5%E6%9C%BA" \o "图灵机)一样可以完成任何计算任务。虽然brainfuck的计算方式如此与众不同，但它确实能够正确运行。

这种语言基于一个简单的机器模型，除了指令，这个机器还包括：一个以字节为单位、被初始化为零的数组、一个指向该数组的指针（初始时指向数组的第一个字节）、以及用于输入输出的两个字节流。

下面是这八种状态的描述，其中每个状态由一个字符标识：

|  |  |
| --- | --- |
| **字符** | **含义** |
| > | 指针加一 |
| < | 指针减一 |
| + | 指针指向的字节的值加一 |
| - | 指针指向的字节的值减一 |
| . | 输出指针指向的单元内容（ASCII码） |
| , | 输入内容到指针指向的单元（ASCII码） |
| [ | 如果指针指向的单元值为零，向后跳转到对应的]指令的次一指令处 |
| ] | 如果指针指向的单元值不为零，向前跳转到对应的[指令的次一指令处 |

（按照更节省时间的简单说法，]也可以说成“向前跳转到对应的[状态”。这两解释是一样的。）

（第三种同价的说法，[意思是"向后跳转到对应的]"，]意思是"向前跳转到对应的[指令的次一指令处，如果指针指向的字节非零。"）

Brainfuck程序可以用下面的替换方法翻译成[C语言](https://zh.wikipedia.org/wiki/C%E8%AF%AD%E8%A8%80" \o "C语言)（假设ptr是char\*类型）：

|  |  |
| --- | --- |
| **Brainfuck** | **C** |
| > | ++ptr; |
| < | --ptr; |
| + | ++\*ptr; |
| - | --\*ptr; |
| . | putchar(\*ptr); |
| , | \*ptr =getchar(); |
| [ | while (\*ptr) { |
| ] | } |

## 例子[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=2" \o "编辑小节：例子)]

### Hello World![[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=3" \o "编辑小节：Hello World!)]

一个在屏幕上打印"Hello World!"的程序：

++++++++++**[**>+++++++>++++++++++>+++>+<<<<-**]**>++**.**>+**.**+++++++**..**+++**.**>++**.**<<+++++++++++++++**.**>**.**+++**.**------**.**--------**.**>+**.**>**.**

### 当前位置清零[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=4" \o "编辑小节：当前位置清零)]

**[**-**]**

### 字符I/O[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=5" \o "编辑小节：字符I/O)]

**,.**

从键盘读取一个字符并输出到屏幕上。

### 简单的循环[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=6" \o "编辑小节：简单的循环)]

**,[.,]**

这是一个连续从键盘读取字符并回显到屏幕上的循环。注意，这里假定0表示输入结束，事实上有些系统并非如此。以-1和"未改变"作为判断依据的程序代码分别是",+[-.,+]"和",[->+>-<<]>[-<+>]>[[-]<<.[->>+<<],[->+>-<<]>[-<+>]>]"。

### 指针维护[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=7" \o "编辑小节：指针维护)]

>**,[.**>**,]**

通过移动指针保存所有的输入，供后面的程序使用。

### 加法[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=8" \o "编辑小节：加法)]

**[**->+<**]**

把当前位置的值加到后面的单元中（破坏性的加，它导致左边的单元被归零）。

### 条件指令[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=9" \o "编辑小节：条件指令)]

**,**----------**[**----------------------**.,**----------**]**

这个程序会把从键盘读来的小写字符转换成大写。按回车键退出程序。

首先，我们通过,读入第一个字符并把它减10（10 在大多数情况下为换行符 LF 的值）。如果用户按的是回车键，循环命令（[）就会直接跳转到程序的结尾：因为这时第一个字节已经被减到了零。如果输入的字符不是换行符（假设它是一个小写字符），程序进入循环。在这里我们再减去剩下的22，这样总共减掉32：这是ASCII码中小写字符和大写字符的差值。

下面我们把它输出到屏幕。然后接收下一个输入字符，并减去10。如果它是换行符，退出循环；否则，再回到循环的开始，减去22并输出……当循环退出时，因为后面已经没有其他的指令，程序也随之终止。

### 加法[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=10" \o "编辑小节：加法)]

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

这个程序对两个一位数做加法，并输出结果（如果结果也只有一位数的话）：

4+3 7

（现在程序开始有点复杂了。我们要涉及到数组中单元的内容了，比如[0]、[1]、[2]之类。）

第一个输入的数字被放在在[0]中，从中减去48来把它从ASCII码值48到57转换为数值0到9：这是通过在[1]中放入6，然后按照[1]中的次数让一个循环从[0]中多次减去8来完成的（当加上或减去一个大的数值时，这是常用的办法）。下一步，加号被读入[1]中；然后，第二个数字被输入，覆盖掉加号。

下面的循环[<+>-]执行最重要的工作：通过把第二个数字移动到第一个里面让它们相加，并把[1]清空。这里的每次循环都把[0]增一并从[1]中减一；最终，在[1]被置零的多次循环中，[1]中的值就被转移到了[0]中。现在，[1]中是我们输入的换行符（这个程序里，我们没有设置对输入错误的检查机制）。

然后，指针被移回到指向[0]，并输出它的内容（[0]里面现在是 a + (b + 48) 的值，因为我们没有修改b的值，这等于 (a + b) + 48，也就是我们想要输出的ASCII值）。然后，把指针指向[1]，里面保存着前面输入的换行符；输出换行符，程序结束。

### 乘法[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=11" \o "编辑小节：乘法)]

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

和前一个程序类似，不过这次是乘法而不是加法。

第一个输入的数字被放入[0]，星号和第二个数字被放入[1]，然后两个数值都被校正：减去48。

现在，程序进入了主循环。我们的基本思想是：每次从[0]中减去一，同时把[1]的值加入到保存乘积的[2]中。在实际操作中，第一个内层循环把[1]的值同时转移到[2]和[3]中，同时[1]清零（这是我们复制数字的基本方法）。下一个内层循环把[3]中的值重新放回到[1]，并清零[3]。然后从[0]中减一，结束外层循环。在退出这个循环时，[0]中为零，[1]仍然是输入的第二个数值，[2]则是这两个数值的和。（要是想保存第一个数，我们可以在外层循环中每次给[4]加一，最后把[4]移回[0]。）

在结果中加48，并把换行符读入[3]，输出ASCII码的乘积，然后输出刚才保存的换行符。

## 注释[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=12" \o "编辑小节：注释)]

* 注意，这里[数组](https://zh.wikipedia.org/wiki/%E6%95%B0%E7%BB%84)的每个单元都是一个字节大小；-命令允许[溢出](https://zh.wikipedia.org/wiki/%E6%BA%A2%E5%87%BA)，它可以用255个[+](https://zh.wikipedia.org/wiki/%2B)命令来代替。同样，如果数组单元是有限、循环的，<可以用29999个>命令代替。每个修改动作都可以被分解为最多7条指令。可是，两个连在一起的修改动作将会破坏“图灵完全”，因为这会把可能的内存状态限制到有限个数。（更确切的说，从这个角度看，现代的计算机依然不是完全意义上的“[图灵完全](https://zh.wikipedia.org/wiki/%E5%9B%BE%E7%81%B5%E5%AE%8C%E5%85%A8)”。）

## 外部链接[[编辑](https://zh.wikipedia.org/w/index.php?title=Brainfuck&action=edit&section=13" \o "编辑小节：外部链接)]

* [Brian Raiter, Muppetlabs. **Brainfuck：八条指令的图灵完全编程语言**](http://www.muppetlabs.com/~breadbox/bf/)。这个网站包括一个brainfuck程序[quine](https://zh.wikipedia.org/wiki/%E8%87%AA%E7%94%A2%E7%94%9F%E7%A8%8B%E5%BC%8F)。
* [Panu Kalliokoski. **Brainfuck档案**](http://esoteric.sange.fi/Brainfuck/)有许多brainfuck实现、程序和quine.
* [Cat's Eye Technologies. **Brainfuck**](http://www.catseye.mb.ca/esoteric/bf/)
* [Frans Faase. **BF is Turing Complete**](http://home.planet.nl/~faase009/Ha_bf_Turing.html)
* [**Brainfucked** - Brainfuck Compiler](http://home.arcor.de/partusch/html_en/bfd.html)