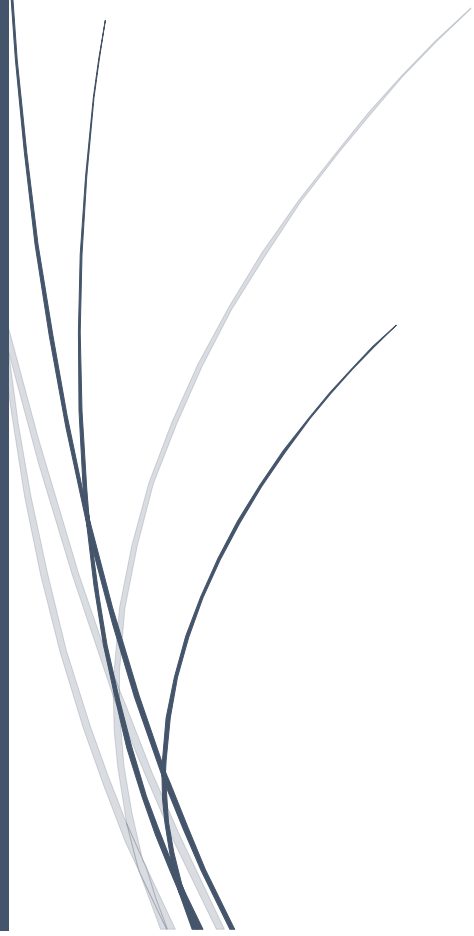
05/10/2016

Brainfuck Level 1

Café Ou Thé

Participants :

- UNG Eric
- GREAUX Thomas
- FERAUD Florian
- DURANDO Fabien



Slice 0

Le slice 0 consistait en la gestion des arguments (-p) et le fait que le fichier existe ou non. Il a aussi fallu créer le script lançant l'exécutable.

Tout d'abord, nous avons créé une classe « ReadFile » permettant la lecture du fichier avec la méthode « reading », celle-ci génère un code d'erreur si le fichier n'existe pas et si l'argument « -p » est rentré.

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p fileMissing.bf
File doesn't exist
```

Si le fichier existe, le code s'exécute sans erreur. La classe contient aussi une méthode « operator » permettant d'effectuer l'opération nécessaire en fonction du mot écrit dans le fichier donné en paramètre.

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p empty.bf
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ echo $?
0
```

Ensuite, dans un script exécutable, on se contente de récupérer les paramètres reçus et de les transmettre au programme précédent.

Slice 1

Cette fonctionnalité Java est développée sous la forme d'une méthode « incr » située dans Memory.java :

- incr : byte -> byte

Cette méthode a pour paramètre la valeur de case mémoire, et retourne la nouvelle valeur de celle-ci, une fois incrémentée.

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p incrBy7.bf
C0: 7
```

Pour cette méthode, on lève une exception si la valeur de la cellule-cible dépasse 255, le code de retour sera alors 1 :

Pour cette méthode, on lève une exception si l'on essaie d'incrémenter la valeur de la cellule-cible au-dessus de 255, le code de retour sera alors 1 :

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p incrByTooMuch.bf
Out of capacity
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ echo $?
1
```

Slice 2

Cette fonctionnalité Java est développée sous la forme d'une méthode « decr » située dans Memory.java :

- decr : byte -> byte

Cette méthode a pour paramètre la valeur de case mémoire, et retourne la nouvelle valeur de celle-ci, une fois incrémentée. Pour ce test, nous avons effectué 7 INCR puis 2 DECR afin de ne pas avoir de cas d'exception.

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p incrBy7AndDecr2.bf
C0: 5
```

Pour cette méthode, on lève une exception si l'on essaie de décrémenter la valeur de la cellule-cible en dessous de 0, le code de retour sera alors 1 :

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p decr.bf
Out of capacity
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ echo $?
1
```

Slice 3

Cette fonctionnalité Java est développée sous la forme d'une méthode « right » située dans Memory.java :

- right : byte

Cette méthode a pour paramètre la position actuelle de l'indice, et va incrémenter cette valeur pour permettre un décalage vers la droite. Dans le cadre de ce test, nous avons utilisé right trois fois, puis incr :

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p rightBy3AndIncr.bf
C3: 1
```

Pour cette méthode, on lève une exception si l'on essaie d'avoir une valeur d'indice supérieure à la taille du tableau (30 000) :

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p rightTooMuch.bf
index impossible
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ echo $?
2
```

Slice 4

Cette fonctionnalité Java est développée sous la forme d'une méthode « left » située dans Memory.java :

- left : byte

Cette méthode a pour paramètre la position actuelle de l'indice, et va incrémenter cette valeur pour permettre un décalage vers la droite. Dans le cadre de ce test, nous avons utilisé right, puis left et enfin incr :

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p rightLeftIncr.bf
C0: 1
```

Pour cette méthode, on lève une exception si la valeur de l'indice devient supérieure à 30000, le code de retour sera alors 2 :

Pour cette méthode, on lève une exception si l'on essaie d'avoir une valeur d'indice inférieure à 0 :

```
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ ./bfck -p left.bf
index impossible
user@user-VirtualBox:~/Documents/cafe_ou_the-tests/Brainfuck$ echo $?
2
```

Keep Calm and Take a Step Back

Nous avons commencé par ses différentes spécifications car ce sont les bases du Brainfuck. En effet, sans ses différentes slices la suite du projet n'est pas possible.

L’ordre n’a pas tellement d’importance, mise à part le slice 0 qui doit forcément être faite en première pour pouvoir implémenter les slices suivants.

Il a été très intéressant de travailler ainsi : en découpant le travail en différentes couches indépendante, chaque membre n’était pas dépendant de l’avancement des autres et pouvait donc avancer à son rythme.