

HomePosition

Inazuma110

1 プログラムの仕様

Brainfuck という言語のインタプリタを作成し、それを元にオリジナルの言語「HomePosition」を作成する。

1.1 Brainfuck とは

Brainfuck はコンパイラがなるべく小さくなるように開発された言語である。可読性、記述性が非常に低い
ため実用性は期待できないが、チューリング完全^{*1}である。

1.2 HomePosition とは

今回作成した Brainfuck の派生言語である。ホームポジションから指を動かす回数が最小限でコーディング
ができることをコンセプトに設計した。Brainfuck の 8 つの各命令を表の通りに置き換えた。

1.3 Brainfuck と HomePosition

今回作成した HomePosition は Brainfuck の 8 つの命令を表 1 のように置き換えた。

表 1 Brainfuck と HomePosition の命令の対応

Brainfuck	HomePosition	命令の意味
>	h	配列のインデックスをインクリメントする。
<	l	配列のインデックスをデクリメントする。
+	j	現在指し示している配列の値をインクリメントする。
-	k	現在指し示している配列の値をデクリメントする。
.	a	現在指し示している配列の値を出力する。
,	s	現在指し示している配列の位置に、入力された値を代入する。
[d	現在指し示している配列の値が 0 ならば対応する] の直後にジャンプする。
]	f	現在指し示している配列の値が 0 でないならば、対応する [の直後にジャンプする。
その他の文字	その他の文字	コメント文として無視される。

^{*1} 計算理論において、ある計算のメカニズムが万能チューリングマシンと同じ計算能力をもつ。一般的なプログラミング言語の多くはチューリング完全である。

2 プログラム

2.1 設計

RunCode クラスでプログラムを 1 文字ずつ解析し、各文字ごとに命令を実行する。Main クラスでは、ファイルから文字列を読み取り、RunCode クラスのインスタンスを生成する。

2.2 Main クラス

String 型変数 sourceCode にコマンドライン引数によって与えられたファイル名のファイルから文字列を読み取る。15 行目の if 文でファイル名が入力されたかを確認し、22 行目の if 文では入力されたファイルが存在するかを確認する。最後に 40 行目で RunCode のインスタンスを、引数ソースコードで生成する。

リスト 1 にプログラムを示す。

リスト 1 Main.java

```
1 import java.io.File;
2 import java.io.FileReader;
3 import java.io.BufferedReader;
4 import java.io.IOException;
5
6 class Main{
7     public static void main(String [] args){
8         // 別ファイルのソースコードを保存するためのString型変数。
9         String sourceCode = null;
10
11         // ファイルを受け取る。
12         try{
13             File file = null;
14             // ファイル名が入力されていないときの処理
15             if(args.length == 1)
16                 file = new File("./" + args[0]);
17             else{
18                 System.out.println("ファイル名の入力が正しくありません。");
19                 System.exit(0);
20             }
21             // ファイルがあるかないかで分岐。
22             if(file.exists()){
23                 FileReader fr = new FileReader(file);
24                 BufferedReader br = new BufferedReader(fr);
25                 String sourceLine;
```

```

26         // ソースコードを読み取る。
27         while((sourceLine = br.readLine()) != null) sourceCode +=
            sourceLine;
28         fr.close();
29     }else{
30         // ファイル名が正しくないときの処理。
31         System.out.println("ファイルが見つかりませんでした。");
32         System.exit(0);
33     }
34     // 例外
35 }catch (IOException ioe) {
36     ioe.printStackTrace();
37 }
38
39 // RunCodeインスタンスを作成。
40 RunCode rc = new RunCode(sourceCode);
41
42 }
43 }

```

2.3 RunCode クラス

フィールドを宣言する。(表 2)

2.3.1 コンストラクタ

コンストラクタで各変数の初期化等を行う。35 行目でソースコードを読み取り、「a」が含まれていた場合、readInput メソッドを実行する。最後に run メソッドを実行する。

2.3.2 run メソッド

for 文でソースコードの文字数文だけ回す。judgeOrder メソッドによって命令を実行する。

2.3.3 judgeOrder メソッド

source の now 番目の文字によって命令を決定する。「j」「k」「h」「l」は表 1 ままの挙動を行う。

「s」の場合、現在指し示している int 型で配列の値を char 型にキャストして出力する。

「a」の場合、入力された文字を Byte で保持している配列から int 型にキャストしたものを受け取る。

「d」の場合、whileIndex に現在地を push し、「f」の場合は配列の値が 0 でなければ whileIndex の先頭位置に戻り、0 なら pop してから処理を続行させる。

2.3.4 readInput メソッド

Scanner クラスで入力を読み込む。102 行目で inputSource に Byte 型に変換した入力値を代入する。
リスト 2 にプログラムを示す。

表 2 宣言するフィールド

変数名	型	説明
source	String	Main クラスから受け取ったソースコードを保持。
index	int	配列のどこを指すかを保持する。0 で初期化。
now	int	ソースコードの何文字目を実行しているかを保持。
inputSource	byte[]	「a」実行の際に、入力された文字を Byte 型で保持。
inputIndex	int	読み取った文字の何文字目を利用するかを保持する。
whileIndex	deque<Integer>	「d」実行時にその位置を保持する。 deque をスタックとして利用することでネストに対応する。
homePositionArray	int[]	実行の際に利用する配列。

リスト 2 Main.java

```
1 import java.util.Scanner;
2 import java.util.Deque;
3 import java.util.ArrayDeque;
4
5 class RunCode{
6     // String型のソースコード
7     private String source;
8     // 現在地。配列の何番目にいるかを表す。
9     private int index = 0;
10    // ソースコードの何文字目を実行しているかを表す。
11    private int now;
12    // 入力された文字列を ASCII文字コードとして10進の数字に変換。
13    private byte[] inputSource;
14    // 読み込んだ文字の何文字目を利用するか。
15    private int inputIndex = 0;
16    // dが入力されたとき、その位置を保持。両端キューをスタックとして利用。
17    Deque<Integer> whileIndex;
18    // 実行の際に利用する配列の初期化。
19    // 配列のインデックスをメモリの値として扱う。
20    int[] homePositionArray = new int[100000000];
21
22
23    /**
```

```

24  * 詠みこんだファイルの実行を行う。
25  * ファイルを読み込み、runメソッドを実行する。
26  * @param s ファイルから読み込んだ文字列。
27  */
28  public RunCode(String s){
29      // Mainクラスより受け取ったファイルの文字列をsourceに保持。
30      source = s;
31      whileIndex = new ArrayDeque<>();
32      // homePositionArrayを0で初期化。
33      for(int i = 0; i < homePositionArray.length; i++) homePositionArray[i]
          = 0;
34      // ソースコードにaが含まれているとき、readInputにより入力を受け取る。
35      if(source.contains("a")) readInput();
36      // プログラム実行
37      this.run();
38  }
39
40  /**
41   * ソースコードを1文字ずつ解析し実行する。
42   */
43  public void run(){
44      // sourceの最初の4文字は"null"なのでそこは飛ばす。
45      for (now = 4; now < source.length(); now++) {
46          judgeOrder(source.substring(now, now+1));
47      }
48  }
49
50  /**
51   * 文字を判定して、文字によって操作を変更する。
52   * @param order ソースコードのnow番目の文字。
53   */
54  public void judgeOrder(String order){
55      switch (order) {
56          case "j":
57              homePositionArray[index]++;
58              break;
59          case "k":
60              homePositionArray[index]--;
61              break;

```

```

62     case "h":
63         index--;
64         break;
65     case "l":
66         index++;
67         break;
68     case "s":
69         System.out.print((char)homePositionArray[index]);
70         break;
71         // 読み取った文字をhomePositionArrayに代入する。
72     case "a":
73         try{
74             homePositionArray[index] = (int)inputSource[inputIndex];
75             inputIndex++;
76         }catch(Exception e){
77             // 処理なし
78         }
79         break;
80         // whileIndexにdの位置を保存。
81     case "d":
82         whileIndex.addFirst(now);
83         break;
84     case "f":
85         // 現在地の
86         homePositionArrayの値が0でなければnowの値をdの位置に変更し、
87         ループ処理をする。
88         if(homePositionArray[index] > 0) now = whileIndex.peek();
89         // 0ならスタックから排除。
90         else whileIndex.removeFirst();
91         break;
92     }
93 }
94 /**
95  * 入力された文字をByte型に変換。
96  */
97 public void readInput(){
98     Scanner sc = new Scanner(System.in);
99     String input = sc.next();

```

```

100     try{
101         // 入力された文字をByte型に変換。
102         inputSource = input.getBytes("US-ASCII");
103     }catch (Exception e){
104         // 処理なし
105     }
106 }
107 }

```

3 実行例

基本的に、出力するには文字コードを参照する必要があるため、図 1 を見てコードを作成する。今回は例として「Hello,World!」と表示されるプログラムを作成する。(リスト 3)

3.1 プログラム

まず、「H」を出力する際に配列の値を 72 にする必要がある。この時、「j」を 72 回入力することで配列の値を 72 にすることも可能だが、今回は「d」と「f」を工夫して用いることにより値を 72 にしている。

まず、配列の値を「j」を 9 回入力することにより 9 にする。次に「d」で繰り返し処理を始め宣言をし、隣の配列の値を順に 8,11,5 まで増やしたあとにはじめにいた位置に戻る。最後にその位置の値を「k」で 1 減らし繰り返しの処理を終了する。こうすることで、8,11,5 まで増やす処理が 9 回行われ、最終的な配列の値は、0, 72, 99, 45 となる。72 が格納されている部分の値を「s」で出力すれば、ターミナルには「H」が出力される。以下の文字も処理の内容としてはほとんど同じである。

なお、文字「h」「j」「k」「l」「f」「a」「s」「d」は命令として実行してしまうため、大文字にすることでコメントとして利用している。実行結果を図 2 に示す。

文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進
NUL	0	00	DLE	16	10	SP	32	20	@	64	40	P	80	50	`	96	60
SOH	1	01	DC1	17	11	!	33	21	A	65	41	Q	81	51	a	97	61
STX	2	02	DC2	18	12	"	34	22	B	66	42	R	82	52	b	98	62
ETX	3	03	DC3	19	13	#	35	23	C	67	43	S	83	53	c	99	63
EOT	4	04	DC4	20	14	\$	36	24	D	68	44	T	84	54	d	100	64
ENQ	5	05	NAK	21	15	%	37	25	E	69	45	U	85	55	e	101	65
ACK	6	06	SYN	22	16	&	38	26	F	70	46	V	86	56	f	102	66
BEL	7	07	ETB	23	17	'	39	27	G	71	47	W	87	57	g	103	67
BS	8	08	CAN	24	18	(40	28	H	72	48	X	88	58	h	104	68
HT	9	09	EM	25	19)	41	29	I	73	49	Y	89	59	i	105	69
LF*	10	0a	SUB	26	1a	*	42	2a	:	58	3a	Z	90	5a	j	106	6a
VT	11	0b	ESC	27	1b	+	43	2b	;	59	3b	[91	5b	k	107	6b
FF*	12	0c	FS	28	1c	,	44	2c	<	60	3c	L	76	4c	l	108	6c
CR	13	0d	GS	29	1d	-	45	2d	=	61	3d	M	77	4d	m	109	6d
SO	14	0e	RS	30	1e	.	46	2e	>	62	3e	N	78	4e	n	110	6e
SI	15	0f	US	31	1f	/	47	2f	?	63	3f	O	79	4f	_	95	5f
															o	111	6f
															DEL	127	7f

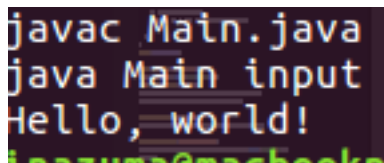
図1 ASC-II 文字コード [1]

リスト3 helloWorld

```

1  j j j j j j j j d l j j j j j j j j l j j j j j j j j l j j j j j h h h k f l s      //H(小文字)
2  l j j s          //E(小文字)
3  j j j j j j s s //LL(小文字)
4  j j j s          //o
5  l k s            //,
6  k k k k k k k k k k s          //W
7  h j j j j j j j s          //o
8  k k k k k k k k s //r
9  j j j s          //L(小文字)
10 k k k k k k s k k k k k k k k s //D(小文字)
11 l j s            //!
12 l j j j j j j j j s          //改行

```



```

javac Main.java
java Main input
Hello, world!

```

図2 helloWorld の実行結果

4 GitHub リポジトリ

https://github.com/Inazuma110/home_position

参考文献

- [1] wikipedia Brainfuck <https://ja.wikipedia.org/wiki/Brainfuck>
- [2] IT 用語辞典 e-Words <http://e-words.jp/p/r-ascii.html>