

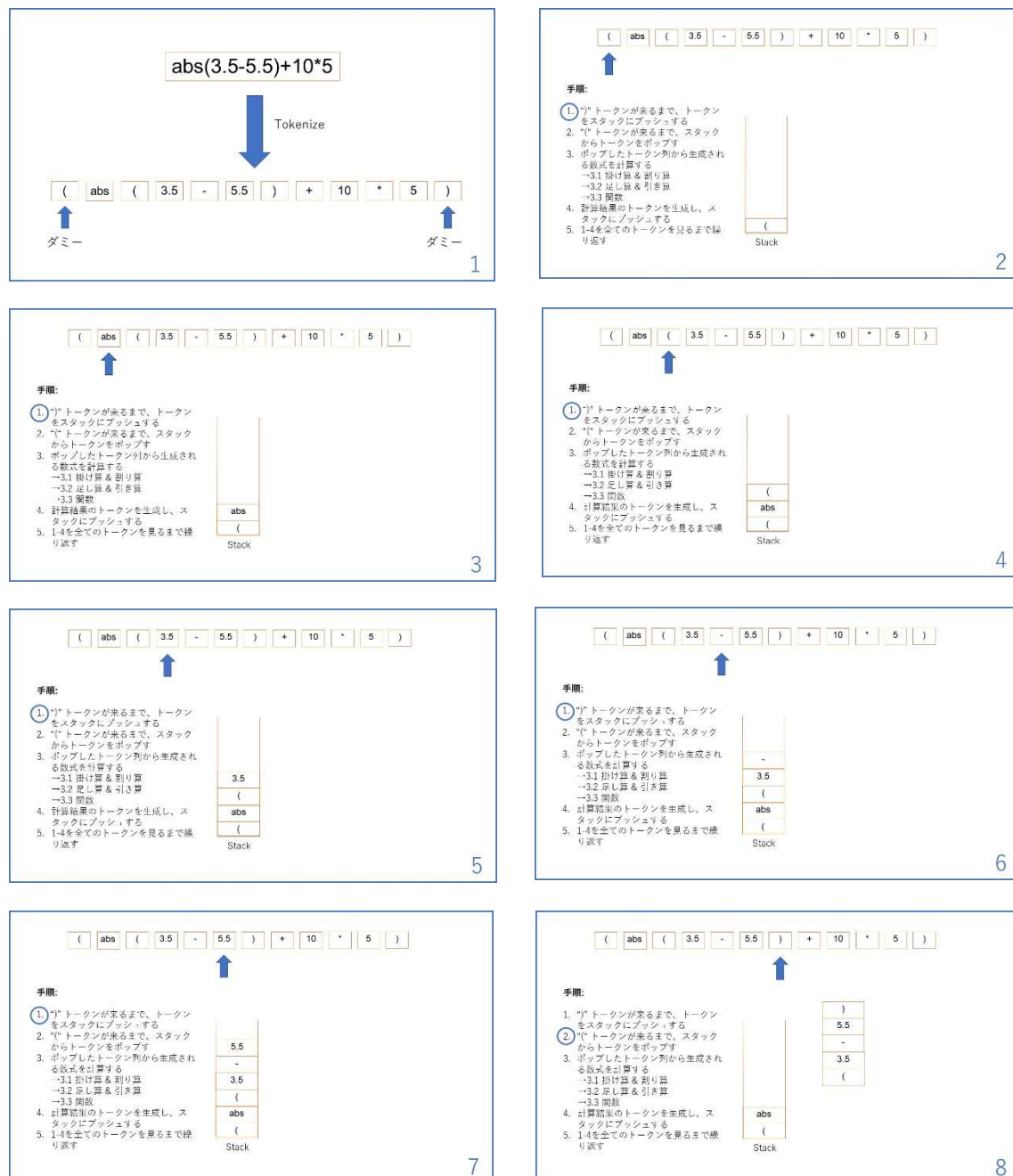
電卓プログラム

横山奈穂

概要

このプログラムは電卓を実装しています。ユーザーが数式を入力すると、プログラムはその値を計算して返します。対象とする演算は、足し算、引き算、掛け算、割り算、括弧の使用、絶対値関数、整数値関数、および四捨五入関数です。

プログラムの仕組み



手順:

- 1.) "トークンが来るまで、トークンをメタツクリープに示す
- 2.) "トークンが来るまで、スタックからトークンをポップする
- 3.) ポップしたトークン列から生成される数式を計算する
 - 3.1 掛け算と割り算
 - 3.2 引き算と引き算
 - 3.3 閉括
- 4.) 計算結果のトークンを生成し、スタックにプッシュする
- 5.) 1-4を全てのトークンを見るまで繰り返す

(

abs

(

3.5

-

5.5

)

+

10

*

5

)

手順:

1. ")" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数値を計算する
 - 3.1 掛けと割り算
 - 3.2 足し算と引き算
 - 3.3 閉括弧
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1.4 を全てのトークンを見るまで繰り返す

-2.0	}	2.0
abs		

関数の計算の場合は、スタックの一番上に関数トークンかどうかを見る

関数トークンならポップして計算

Stack

手順:

1. "j" トークンが来るまで、トークンをスタックにプッシュする
2. "i" トークンが来るまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数式を計算する
 - 3.1 掛け算と割り算
 - 3.2 足し算と引き算
 - 3.3 閉括弧
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1.4 を全てのトークンを見るまで繰り返す

(

abs

(

3.5

-

5.5

)

+

10

*

5

)

手順:

- ① "I" トークンが来るまで、トークンをスタックにプッシュする
- "I" トークンが来るまで、スタックからトークンをポップする
- ポップしたトークン列から生成される数値を計算する
 - 3.1 掛け算 & 割り算
 - 3.2 足し算 & 引き算
 - 3.3 乗数
- 計算結果のトークンを生成し、スタックにプッシュする
- 1.4 を全てのトークンを見るまで繰り返し実行

+
2.0
(

Stock

手順:

1. "a" トークンが来るまで、トークンをスタックにプッシュする
2. "(" トークンが来るまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数式を計算する
 - 3.1 掛け算と割り算
 - 3.2 足し算と引き算
 - 3.3 閉放
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1.4 を全てのトークンを見るまで繰り返す

手順:

- 1) "d" トークンが来るまで、トークンをスタックにプッシュする
- 2) "d" トークンが来るまで、スタックからトークンをポップする
- 3) ポップしたトークン列から生成される数式を生成する
 - 3.1 掛け算 & 割り算
 - 3.2 引き算 & 引き算
 - 3.3 閉括弧
- 4) 計算結果のトークンを生成し、スタックにプッシュする
- 5) 1-4 を全てのトークンを見るまで繰り返し直す

手順:

1. "j" トークンが出るまで、トークンをスタックにプッシュする
2. "i" トークンが出るまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数式を計算する
 - 3.1 掛け算と割り算
 - 3.2 足し算と引き算
 - 3.3 閉括弧
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す

手順:

1. “)”トークンが来るまで、トークンをスタックにプッシュする
2. “)”トークンが来るまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数式を計算する
 - 3.1 掛け算と割り算
 - 3.2 加減算と引き算
 - 3.3 乗数
4. 計算結果のトークンを生成し、スタックにプッシュする

14番全てのトークンを見るまで繰り返す

Stack

手順:

1. "3"トークンが来るまで、トークンをスタックにプッシュする
2. "("トークンが変わるまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数値を計算する
 - 3.1 掛け算と割り算
 - 3.2 足し算と引き算
 - 3.3 閉括
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す

stack	token
/	/
/ *	*
/ * +	+
/ * + *	*

手順:

1. ")"トークンが来るまで、トークンをスタックにプッシュする
2. "*)"トークンが来るまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数式を計算する
 5.5 * 3.5 = 32.5 (乗算)
4. 計算結果のトークンを生成し、スタックにプッシュする
5. "+"を全てのトークンを見るまで繰り返す

手順:

1. ")"トークンが来るまで、トークンをスタックにプッシュする
2. "("トークンが来るまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算 & 割り算
→3.2 足し算 & 引き算
→3.3 四則
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す

Stack: 52.0

19

手順:

1. ")"トークンが来るまで、トークンをスタックにプッシュする
2. "("トークンが来るまで、スタックからトークンをポップする
3. ポップしたトークン列から生成される数式を計算する
→3.1 掛け算 & 割り算
→3.2 足し算 & 引き算
→3.3 四則
4. 計算結果のトークンを生成し、スタックにプッシュする
5. 1-4を全てのトークンを見るまで繰り返す

Stack: 52.0

Answer!

20

入力の対象とするもの・しないもの

対象

- ・ 負の値 (例. $-1+5$ は -6 と計算される)
- ・ 必要以上に括弧のペアを使用する (例. $(((((5+5))))))$ の計算結果は $5+5$ と同様)
- ・ 括弧のペアのうち、“(“を多く使用する (外側の括弧が無視される、例. $(3+5*2)$ は $3+5*2$ となり答えは 13 となる)

→ただし、電卓プログラムが正しい答えを返すか判定する"eval" 関数はエラーを返す

対象ではない

- ・ ゼロ除算
- ・ 数式中に空白を入れる (例. $1 + 5$)
- ・ 不完全な数の入力 (例. $.5$)
- ・ 括弧のペアのうち、")"を多く使用する (例. $3+5)$)