コンパイラ理論 9
Racc その2 練習問題

櫻井彰人

Racc はパーサージェネレータ

- コンパイラコンパイラは、意味（アクション）もコンパイラコンパイラ記述言語で書ける（べき）。
- しかし、RaccはRubyで書き、yaccはCで書く。
- これは、Raccやyaccは（Bisonも…も）「パーサージェネレータ」であるということを意味する

Racc の練習問題1

- calc.y を改造して、べき乗ができるようにして下さい。べき乗演算子は、一文字の "^" としましょう。
  実行すると

```
$ ruby calcp.rb

type "Q" to quit.


? 2^3
= 8

?
```

```
? 2^3^4
= 2417851639229258349412352

? -2^(3^4)
= -2417851639229258349412352

? 4^2^3
= 65536

? -4^(2^3)
= -65536

?
```

分かりにくいところ

^
　行頭。文字列の先頭や改行文字の直後の位置にマッチします。
¥A
　文字列先頭。^ とは異なり改行の有無には影響しません。

¥s
　空白文字。[¥t¥n¥r¥f] と同じ

　¥1, ¥2 ... ¥*n*
　　後方参照(back reference)。

　　後方参照
　　n 番目の括弧(正規表現 ( ) グルーピング)にマッチした文字列にマッチします

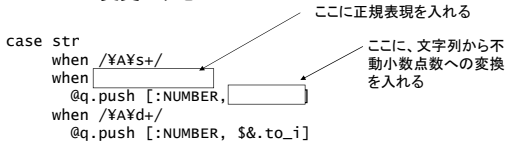
　例: show_regexp("Hello to the the world", /(¥w+)¥1/)

べき乗の導入へのヒント

- 2項演算子なので、他の2項演算子と同様に考えればよい
- 結合度（優先順位）は、*/ より上位、単項演算子よりも上位
  - -2^4 = - (2^4)
- 右結合（right associative）である
  - 2^3^4 = 2^(3^4)

Raccの練習問題2

- 浮動小数点数が扱えるようにする。
- Ruby は変数に型がないため、そして、今作っている電卓にも型がないため、数値表現のみを可能とすれば、よい
- scannerの変更ですむ



Ruby 補足

- **配列: 要素の追加・削除**
- push メソッドを使用して、配列に要素を追加することができる。

```
irb(main):001:0> a = [1,2,3,4]
=> [1, 2, 3, 4]
irb(main):002:0> a.push(10)
=> [1, 2, 3, 4, 10]
irb(main):003:0>
```

- pop メソッドを使用して、配列の最後の要素を取り出すことができる

```
irb(main):003:0> a.pop
=> 10
irb(main):004:0> a
=> [1, 2, 3, 4]
irb(main):005:0>
```

Ruby 補足: ハッシュ

- 配列では、インデックスを用いて要素を指定する。ハッシュでは、キーと呼ばれるものを用いて要素を指定する。

```
irb(main):001:0> h = Hash::new
=> {}
irb(main):002:0> h['apple'] = 150
=> 150
irb(main):003:0> h['banana'] = 200
=> 200
irb(main):004:0> h['lemon'] = 300
=> 300
irb(main):005:0> h
=> {"apple"=>150, "banana"=>200, "lemon"=>300}
irb(main):006:0> h['apple']
=> 150
irb(main):007:0>
```