sound - チュートリアル

目次

1. sound とは

sound は、声に出しながら書くと音が可愛いプログラミング言語である。この言語は音にこだわりがあるため、sound と名付けた。作者はコードをよく声に出しながら書いており、より楽しくコードを書くことができる言語は、繰り返し文や出力文を keepon（キィポン）や visu（ビジュ）のような短い単語かつ、声に出してテンションが上がるものだと面白いと考えた。

sound は、手続き型言語の基本機能である逐次実行と para 文による分岐、keepon 文による繰返し、複数の式をまとめるためのブロック化の機能の 4 つの機能のみを持つ。また、入出力は、get 文、visu 文として表される。変数や代入文の使用は可能である。拡張子は「.on」である。

2. 構文ルール

- 文列 = 文（文）*
- 文 = 代入文 | para 文 | keepon 文 | visu 文 |'{' 文列 '}'
- 代入文 = 変数 '=' (式 | 変数 | 'get' | 真偽値)
- para 文 = 'para' 式 'open' 文 'real' 文 'close'
- keepon 文 = 'keepon' 式 'open' 文 'close'
- visu 文 = 'visu' (変数 | リテラル | 'get')
- 式 = /* calc の式に準ずる */
- 項 = 因子 (('*' | '/') 因子) *
- 因子 =リテラル | 変数 | '(' 式 ')'
- get = 'get' (変数 | リテラル | 真偽値)
- 真偽値 = 'true' | 'false'

構文ルール

sound では、Ruby で言う「代入文」「if 文」「while 文」「print 文」「標準入力」「四則演算」「真偽判定」の処理ができる。その他には、演算子「==」「<」「>」「<=」「>=」の使用も可能である。

変数は、英字のみとなっており、「_」などの記号や空白、数値は使用できない。そのため、「hello world」と出力したい場合は、visu 文で hello と world をそれぞれ出力する必要がある。「visu hello world」と書いた場合は、hello のみが出力される。

数値は、プログラム上も get での標準入力時も「-」（マイナス）の数値は使用できない仕様となっている。しかし、計算によってマイナスになることはエラーとならない。

また、ファイルの最後には空行を入れることもルールとする。「close」の文字や「visu 1」の数値のみを出力するコードが最終行にきた場合、最終行の一つ手前までしか読み取られず、エラーが生じる可能性があることを頭に入れておく必要がある。

3. サンプルコード

サンプルコードの実行には、sound.rb ファイルが必要である。実行は、ターミナルにて「ruby sound.rb ファイル名.on」で可能である。

3-1. 組み合わせの計算（nCr.on）

こちらは、組み合わせの計算を実装した sound プログラムである。標準入力では、n の値、r の値の順に入力が求められる。入力が求められたら、一つ数値を打ち、エンターキーを押すことで次の処理へ進む。実行結果は以下のようになる。

```
% ruby sound.rb nCr.on
nCr
n
5
r
3
AnswerIs
10.0
```

実行結果

ここからは、サンプルコードの解説を行う。1 行目で visu を用いているため nCr を出力する。nCr は変数ではないため、そのまま文字列が出力される。2 行目も同様に n が出力される。3 行目では、get を使用しているため、入力が受け付けられ、nn という変数に格納される。5 行目は、変数ではないため r が出力され、6 行目では変数 rr に入力されたものが格納される。

8 行目から 13 行目が nCr の分子の計算をしている。10 行目の keepon 文では、j の値が nn の値以下であった場合、11 行目と 12 行目の処理が行われる。

15 行目から 20 行目は、nCr の分母の計算をしている。17 行目の keepon 文では、rr の値が i の値以上であった場合、18 行目と 19 行目の処理が行われる。

22 行目で、分子割る分母の計算をしている。また、25 行目の para 文では入力された数値が n の方が r よりも大きいかを確かめ、その判定によってエラー文を出力するか計算結果を出力するかを決めている。

```
 1 visu nCr
 2 visu n
 3 nn = get
 4
 5 visu r
 6 rr = get
 7
 8 Numerator = 1
 9 j=nn-rr+1
10 keepon j<=nn open
11     Numerator = Numerator * j
12     j=j+1
13 close
14
15 Denominator = 1
16 i=2
17 keepon rr>=i open
18     Denominator = Denominator * i
19     i=i+1
20 close
21
22 nCr = (Numerator/Denominator)
23
24
25 para rr > nn open
26     visu error
27     visu r
28     visu LessThan
29     visu n
30 real
31     visu AnswerIs
32     visu nCr
33 close
34
```

nCr.on のソースコード

3-2. 本日の運勢（Fortunes.on）

こちらは、本日の運勢を実装した sound プログラムである。標準入力では、name の値、num の値の順に入力が求められる。入力が求められたら、文字列または数値を打ち、エンターキーを押すことで次の処理へ進む。実行結果は以下のようになる。

```
% ruby sound.rb Fortunes.on
MyNameIs
atsuri
Hi
atsuri
LetsTellFortunes
EnterNumber
5
YouAreLucky
Bye
```

実行結果

ここからは、サンプルコードの解説を行う。1 行目で visu を用いているため MyNameIs を出力する。MyNameIs は変数ではないため、そのまま文字列が出力される。2 行目では、get を使用しているため、入力が受け付けられ、name という変数に格納される。5 行目は、変数 person に変数 name の値を格納しており、6 行目で変数 person の値を出力している。

8 行目と 9 行目もそれぞれ文字列を出力し、10 行目で変数 num に入力された値が格納されている。また、12 行目と 13 行目では真偽値を格納している。

15 行目から 19 行目で本日の運勢を判定しており、22 行目から 28 行目で判定結果を出力している。今回は、変数 num には数値があることを前提としている。

最後に Bye という文字列を出力することで、実行が終了したことを表している。

```
 1 visu MyNameIs
 2 name = get
 3
 4 visu Hi
 5 person = name
 6 visu person
 7
 8 visu LetsTellFortunes
 9 visu EnterNumber
10 num = get
11
12 VeryLucky = false
13 Lucky = false
14
15 para num < 5 open
```

```
 16     VeryLucky = true
 17 real
 18     Lucky = true
 19 close
 20
 21
 22 para VeryLucky == true open
 23     visu YouAreVeryLucky
 24 close
 25
 26 para Lucky == true open
 27     visu YouAreLucky
 28 close
 29
 30 visu Bye
 31
```

Fortunes.on のソースコード

以上