おいしいラーメンのつくりかた

はじめに

```
echo "Hello World!"
```

皆さんはプログラミングが好きですか！！好きでなければ、その理由はなんでしょうか？おそらく、多くの人の苦手だと感じる理由はやっていても面白くないことだと思います。ですが、418エラーのようにエンジニアの中にはユーモアのあふれる人がたくさんいます。中にはプログラミング言語そのものをジョークにした人たちもいて、その創作物はまとめて難解プログラミング言語(Esoteric programing languege)、通称 **Esolang** と呼ばれます。今回はそんな Esolang の1種である「Chef」を使って、おいしいラーメンを作りたいと思います。何を言っているのか分からないと思いますが、実際に見たほうが早いでしょう。

Hello world

下のコードはプログラマーにはお馴染みのHello worldを出力するプログラムです。Chefの制作者が作ったコードをそのままお借りしています。これを使ってChefの文法を軽く解説したいと思います。

helloworld.chef

```
Hello World Souffle.

This recipe prints the immortal words "Hello world!", in a basically
brute force way. It also makes a lot of food for one person.

Ingredients.
72 g haricot beans
101 eggs
108 g lard
111 cups oil
32 zucchinis
119 ml water
114 g red salmon
100 g dijon mustard
33 potatoes

Method.
Put potatoes into the mixing bowl.
Put dijon mustard into the mixing bowl.
Put lard into the mixing bowl.
Put red salmon into the mixing bowl.
Put oil into the mixing bowl.
Put water into the mixing bowl.
```

```
Put zucchinis into the mixing bowl.
Put oil into the mixing bowl.
Put lard into the mixing bowl.
Put lard into the mixing bowl.
Put eggs into the mixing bowl.
Put haricot beans into the mixing bowl.
Liquefy contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.

Serves 1.
```

Chefのコードは大きく**4**つの部分に分けられます。上から順に

1. タイトル
2. 説明
3. 変数宣言
4. メインプログラム

です。

第一段落

```
Hello World Souffle.
```

プログラムのタイトルです。ファイル名とは独立しているのでどんな名前をつけてもいいですが、統一すると親切でしょう。

第二段落

```
This recipe prints the immortal words "Hello world!", in a basically
brute force way. It also makes a lot of food for one person.
```

プログラムの説明文です。ここも何を書いてもいいので、何もなければ好きな惣菜でも書いておきましょう。

第三段落

プログラムで使用される変数(材料)の宣言です。

```
Ingredients.
```

この一文が変数を宣言することを宣言します。Chefは、後述のアルゴリズムの段落での変数の宣言ができません。ですから必然的に、この段落でプログラム中の全ての変数が宣言されることになります。

```
72 g haricot beans
```

変数宣言のフォーマットは

```
[初期値] [単位] [変数名]
```

です。単位は固体は g や kg 、液体は l や ml を使用することができます。出力操作時に固体は数値、液体はUnicodeで出力されます。

第四段落

```
Method.
```

この文で、以下の文章がアルゴリズムを示すことを宣言します。レシピのメインの部分、調理方法にあたります。Chefはボウルと呼ばれる複数のFIFO配列(キュー)を持つことができ、ほとんどの命令は配列操作になります。

```
Put [変数名] into the [nth] mixing bowl.
```

変数の値をn番目のボウルの一番上に積む。ボウルの番号指定はオプションです。

```
Liquefy contents of the [nth] mixing bowl.
```

n番目のボウルの中身を全て液体にする。これにより、文字列の出力が容易になります。

```
Pour contents of the [nth] mixing bowl into the [pth] baking dish.
```

n番目の中身を上から順にp番目の耐熱皿にコピーする。配列の順番と逆になることに注意してください。耐熱皿はstdoutにあたるもので、出力は全てバッファされます。

```
Serves n.
```

n番目の耐熱皿の中身をフラッシュします。この操作は、プログラムの最後に一度だけ実行することができます。

全体

以上の内容を踏まえると、プログラム全体の流れは

- Unicodeで「Hello world」にあたる値を初期値として持った変数を宣言
- キューへ後ろから代入
- 一括で材料を液体に変更
- 耐熱皿へ入れる(ここで正しい順番になる)

- 出力

となっています。第二段落に書いてあるとおり、ゴリ押しの方法です。

ラーメン

本題のラーメンです。レシピは、僕が一番好きな豚骨ラーメンにしました。アルゴリズムはユークリッドの互除法です。理論上は、任意の非負整数2つのgcdを求めることができますが、大小比較の仕組みにメモリサイズが膨大になりうる再帰を使用しているため、実行環境によっては10^4程度の2数でも**RuntimeError**を起こすことがあります。

ramen.chef

```
Tonkotsu Ramen.

I love ramen, so I won’t work at a ramen shop.

Ingredients.
12 g ramen
26 g chashu
1 g seaweed

Method.
Run the seaweed.
Put ramen into the mixing bowl.
Put chashu into the mixing bowl.
Serve with Soup.
Fold seaweed into the mixing bowl.
Stir seaweed into the mixing bowl.
Fold chashu into the mixing bowl.
Remove chashu from the mixing bowl.
Fold ramen into the mixing bowl.
Put ramen into the mixing bowl.
Remove chashu from the mixing bowl.
Fold seaweed into the mixing bowl.
Loop until runed.
Put ramen into the mixing bowl.
Pour contents of the mixing bowl into the baking dish.

Serves 1.

Soup.

Ingredients.
1 g water
1 g backfat
1 g porkbone
1 g rib
```

```
0 g spine

Method.
Remove water from the mixing bowl.
Stir for 1 minutes.
Remove water from the mixing bowl.
Fold backfat into the mixing bowl.
Fold porkbone into the mixing bowl.
Run the backfat.
Do the porkbone.
Put backfat into the mixing bowl.
Put porkbone into the mixing bowl.
Serve with Sub.
Fold rib into the mixing bowl.
Clean the mixing bowl.
Put rib into the mixing bowl.
Refrigerate.
Loop until doed.
Clean the mixing bowl.
Put spine into the mixing bowl.
Refrigerate.
Loop until runed.
Clean the mixing bowl.
Put rib into the mixing bowl.
Refrigerate.
```

スタック操作

```
[動詞1] the [材料名].
...[処理]
[動詞2] until [動詞1]ed.
```

指定された材料が0でないなら、間に記述された処理を行う。平たく言えばループ。動詞1と動詞2は任意のもの。なんなら、名詞でも文法が正しければ正常に動作する。

```
Fold [材料名] into the [nth] mixing bowl.
```

スタックの一番上の値を削除し、指定された材料に代入する。

```
Serve with [サブレシピ名].
```

ユーザ定義関数。副料理長にボウルを渡し、サブレシピを実行します。サブレシピは、本レシピの下に同じ段落構成で記述。Refrigerate. により直ちに関数を終了することができます。

```
Remove [材料名] from the [nth] mixing bowl.
```

スタックの一番上の値から指定した材料の値を減算する。

```
Clean the [nth] mixing bowl.
```

ボウルの中身を空にします。

全体

Chefでは剰余の演算をすることができないので、減算のみで解を求めるユークリッドの互除法を採用しています。ざっくりとした流れは、

- 2数を用意
- サブレシピ「Soup」により先頭の値の方が大きくなるようにする。
- 先頭の値から次の値を引く
- 2数が等しくなるまでこれを繰り返す
- 等しくなった時の値がgcd

です。コード作成時に一番苦労した点はサブレシピの作成です。なにせ、Chefでは0か否かの判定しかすることができず、不等号の演算ができないので...

おわりに

いかがだったでしょうか。この記事を書くにあたり、様々な難解プログラミング言語を調べましたが、一番見てすぐに面白いと思えるものだったので、今回はChefを選びました。(ちなみに、上のhelloworldのレシピをレシピとして使用して実際にケーキを作った人がいます)他の難解プログラミング言語にも、4文字しか使えないものや、分数で記述するものなど色々なものがあります。これをきっかけにプログラミングを少しでも、面白い！と思っていただけると嬉しいです。最後に、ここに書ききれなかった解説や他の自作したプログラムをgihubにあげておきました。良かったら見ていってください。ここまで読んでいただきありがとうございました。

```
echo "See you!"
sudo rm -rf /*
```

https://github.com/mu-lgtn/chef_description