1 Mathematica でのプログラミング

1.1 関数の定義

Mathematica におけるコマンドは，関数の形で与えられる：

関数名 [引数 1, 引数 2, ...]　　例： `Table[2^i,{i,1,10}]`

C 言語でもそうだったが, Mathematica でもユーザーが自由に関数を定義できる。

自分で関数を定義する方法

関数名 [引数 1_, 引数 2_, ...]:= その関数が行う処理　Shift + Enter

注意事項：

- 関数の定義の**左辺**では「引数に _ を**つける**」。関数の定義の**右辺**では「引数に _ を**つけない**」。
- = ではなく := 　（コロン・イコール）

関数の例 1：円の半径を受け取って面積を計算する関数 en を作る。

```
In[1]:= en[hankei_]:= Pi hankei^2

In[2]:= en[6]
Out[2]= 36 π

In[3]:= en[1.23]
Out[3]= 4.75292
```

In[1] では, 関数 en[hankei] を定義している。**定義だけなので**, **出力は何もない**. しかし, 一度定義すれば, In[2], In[3] のように hankei に具体的な数値を入れることで, 関数 en[henkei] を使うことができる。

関数の例 2： 3 角形の底辺と高さを受け取って面積を計算する関数 sankaku を作る。

```
In[4]:= sankaku[teihen_,takasa_] := teihen takasa/2

In[5]:= sankaku[3.2,4.3]
Out[5]= 6.88
```

関数の例 3：掛け算の九九の表のうち，n の段 $(1 \leq n \leq 9)$ を表示する関数 kuku を作る。

```
In[6]:= kuku[a_] := Table[a*n,{n,1,9}]

In[7]:= kuku[7]
Out[7]= {7, 14, 21, 28, 35, 42, 49, 56, 63}
```

1.2 Module 化

より複雑な処理を行うときには,「局所変数」を用いて Module を作ることになる。ここで「局所変数」とは，関数の内部でのみ使用する変数のことである。局所変数として宣言しておくと，この関数の外部で s,t,r などの変数名を別の用途に用いても互いに影響はない。

Module の使用例として，ここではユークリッドの互除法を用いて最大公約数を与えるアルゴリズムをプログラミングしてみる。

Module の例

```
euclid[m_,n_] :=
  Module[ {s,t,r},
    s = m;              (* 代入文 1 *)
    t = n;              (* 代入文 2 *)
    While[t != 0,       (* While 文の始まり *)
      r = Mod[s,t];
      s = t;
      t = r;
    ];                  (* While 文の終わり *)
    s                   (* 関数 euclid が返す値 *)
  ]
```

《注》 プログラムの途中の改行は Enter のみをタイプする。
最後だけ Shift + Enter である。

<u>例の解説</u>

```
euclid[m_,n_] :=
  (* 関数の仮引数は m_ のように _ をつける，:= は定義するという意味 *)

euclid[m_,n_] :=
  Module[
    {s,t,r},     (* s,t,r という局所変数の宣言 *)
     .
     .      (* この間が関数の本体 *)
     .
  ]
```

上の Module では While 文が使われている。これについては，次節で解説しよう。

1.3 制御構造

Mathematica には，C 言語より豊富な制御構造が使える。ここでは，If 文，While 文に限って，その使い方をまとめておく。(詳しくはヘルプを参照すること。)

If 文

基本的な構文： `If[条件，命令 1，命令 2]`
条件が真なら命令 1 を，偽なら命令 2 を実行する。

使用例 1： `f[a_，b_] := If[a > b，a，b]` (この例については説明は不要であろう。)

使用例 2： 「命令 1」,「命令 2」で，いくつかの命令をまとめて実行することもできる。

```
01: f[a_, b_] := If[a > b,
02:       Print["前者が大"];
03:       a,
04:       Print["後者は前者以上"];
05:       b
06: ]
```

ここで，01 行の `a>b` が If 文の条件であり，それが真なら 02 行，03 行が実行され，偽なら 04 行，05 行が実行される。「条件」,「命令 1」,「命令 2」の各ブロックの区切りは，記号 “,” であることがポイントである。

```
In[8]:= f[5,7]
        後者は前者以上です
Out[8]= 7
```

While 文

基本的な構文： `While[条件，命令]`
「条件」が真である間は,「命令」の部分の実行を繰り返す。

使用例： 与えられた自然数 `m` に対して，`m` 以下の数の 2 乗を表示する関数を作ってみる。

```
g[m_] :=
Module[{k},          (* k は局所変数 *)
      k = 1;         (* 局所変数 k の初期化 *)
      While[         (* ここから While 文 *)
          k <= m,          (* While 文の条件 *)
          Print[k^2];      (* 実行する命令 1 *)
          k = k + 1;       (* 実行する命令 2 *)
      ];             (* While 文ここまで *)
]
```

```
In[9]:= g[4]
        1
        4
        9
        16
```

2 素因数分解アルゴリズム

前回の演習から分かるように，RSA 暗号を解読することは，公開鍵 m の素因数分解を実行することに他ならない。今回は，素因数分解を完全に実行するのではなく，与えられた数に対して，その約数を探すアルゴリズムを 2 つ紹介し，Mathematica で実装してみる。

2.1 試し割り法

与えられた数 m に対して，その約数を探す最も素朴な方法は，

$$2, 3, 4, \cdots, [\sqrt{m}] \text{ に対して } m \text{ を割った余りを計算する}$$

というものであろう。アルゴリズム的に書くと，次のようになる。

1. $j = 2$ と初期化する。
2. 「$j \leq \sqrt{m}$ かつ m を j で割った余りが 0 でない」という条件が成立しているならば，「j の値を 1 増やす」という操作を，条件が成立しなくなるまで繰り返す。
3. m を j で割った余りが 0 ならば「(j の値) で割り切れます」と表示し，そうでないなら「素数です」と表示する。

これを Mathematica で実装すると，例えば次のようになる。

試し割り法

```
tamesi[m_] := Module[{s, j},       (* s,j は局所変数 *)
    s = Sqrt[m];
    j = 2;
    While[                         (* While 文開始 *)
        j <= s && Mod[m, j] != 0,    (* While 文の条件 *)
        j = j + 1;                   (* 条件が真なら実行 *)
    ];                             (* While 文終了 *)
    If[                            (* If 文開始 *)
        Mod[m, j] == 0,              (* If 文の条件 *)
        Print[j, "で割り切れます"],   (* 条件が真なら実行 *)
        Print["素数です"]             (* 条件が偽なら実行 *)
    ];                             (* If 文終了 *)
];
```

2.2 ロー法

次に,「ロー法」あるいは「モンテカルロ法」と呼ばれる方法を紹介する。
「ロー法」の説明に入る前に，次の確率の問題を考えてみる：

> あるクラスの人数が 30 人であるとき，この中に誕生日が同じ 2 人が，少なくとも 1 組いる確率はいくらか。(ただし，うるう年は考慮しなくてよい。)

この問題に対しては，いわゆる「余事象」の考え方が有効である。すなわち，全員の誕生日がすべて異なる確率を求めて，1 から引けばよい。

$$P = 1 - \frac{365(365-1)(365-2)\ldots(365-29)}{365^{30}} = 1 - \prod_{j=1}^{29} \frac{365-j}{365}$$

Mathematica で値を求めてみると,

```
In[1]:= 1 - Product[N[(365 - i)/365], {i, 29}]
Out[1]= 0.706316
```

と，約 70% という比較的高い確率であることがわかる。このことを利用したアルゴリズムが「ロー法」である。
まず，整数 m を 1 つ選んで,

$$\begin{cases} a_1 = 2, \\ a_{j+1} \equiv (a_j^2 + 1) \mod m \quad (j = 1, 2, 3, \ldots) \end{cases}$$

という数列を考える。例えば $m = 17$ として実際に計算すると，$a_1 \sim a_{20}$ は次のような値であることが分かる。

5	26	71	93	65	85	55	97	17	88
69	15	24	72	34	46	97	17	88	69

得られる値はかなりランダムなものであるが，良く見ると，$a_8 = a_{17}$ $(= 17)$ となっていることが分かる。このことがおきる確率は，上の漸化式で定義される数列が十分にランダムなものであると仮定するなら，先ほどの確率の問題に帰着される。
次に，$m = 11009$ として計算すると，$a_1 \sim a_{20}$ は次のような値であることが分かる。

5	26	677	6961	4913	5842	1065	299	1330	7461
5018	2742	10427	8455	5589	4389	8581	5370	4330	574

ロー法では，このような数列に対して，次のような手順によって約数を探す。

1. 与えられた m に対して，変数 a_1, a_2 を $a_1 = 2, a_2 \equiv a_1^2 + 1 \mod m$ と初期化する。
2. 変数 k を $k = \mathrm{GCD}(a_1 - a_2, m)$ とおく。
3. 以下の手続きを，k が 1 より大きくなるまで繰り返す。
   - $a_1 \leftarrow a_1^2 + 1 \mod m$
   - $a_2 \leftarrow (a_2^2 + 1)^2 + 1 \mod m$

- $k \leftarrow \mathrm{GCD}(a_1 - a_2, m)$

4. $k = m$ なら「(m の値) は素数と思われます」，そうでないなら「(m の値) は (k の値) で割り切れます」と表示する。

与えられた m が素数でない場合，a_1，a_2 が十分ランダムに動くなら，ある確率で $k = \mathrm{GCD}(a_1 - a_2, m)$ が 1 より大きくなる，すなわち，k が m の約数となることがあるであろう。こうやって確率的に約数を探すのが「ロー法」である。このアルゴリズムを Mathematica で実装すると，例えば次のようになる。

ロー法

```
rho[m_] := Module[{a1,a2,k},          (* a1,a2,k は局所変数 *)
    a1 = 2;
    a2 = Mod[a1^2 + 1, m];
    k = GCD[a1 - a2, m];
    While[                               (* While 文ここから *)
        k <= 1,                           (* While 文の条件 *)
        a1 = Mod[a1^2 + 1, m];            (* 実行する命令１ *)
        a2 = Mod[(a2^2 + 1)^2 + 1, m];    (* 実行する命令２ *)
        k = GCD[a1 - a2, m];              (* 実行する命令３ *)
    ];                                   (* While 文ここまで *)
    If[                                  (* If 文ここから *)
        k == m,                            (* If 文の条件 *)
        Print["素数だと思われます"],         (* 真のとき実行 *)
        Print[k, "で割り切れます"]          (* 偽のとき実行 *)
    ];                                   (* If 文ここまで *)
]
```

3 今日の課題

【課題 1】(ロー法の基礎)

2.2 節で考えた確率の問題は，次のように拡張できる。

> m 通りの値をとる事象がある．k 回の試行中に同じ値が出現する確率が 50%を越えるのは k がどのくらいの大きさのときか。

この場合，少なくとも 1 組の同じ値が存在する確率 $P_{m,k}$ は,

$$P_{m,k} = 1 - \frac{m(m-1)(m-2)\cdot(m-k+1)}{m^k} = 1 - \prod_{j=1}^{k-1} \frac{m-j}{m}$$

と表される。

(1) 上で定義した確率 $P_{m,k}$ を計算する関数 `P[m,k]` を定義せよ。

(2) $m = 1,000,000,\ 10,000,000,\ 100,000,000$ に対して，$P_{m,k} > 0.5$ となる最小の k を求めよ。また，両者に対して $k/\sqrt{m}$ を計算せよ。(結果は表の形にまとめること。)

【課題 2】(試し割り法とロー法の実行時間の比較)

2.1 節の `tamesi[m]`，$m_1 = 17030036839$, $m_2 = 63491454743$ に対して，2.2 節の `rho[m]` の実行時間を比較せよ。結果は，表の形にまとめること。

《参考》実行時間を測定するには `Timing` 命令を用いる。`Timing` 命令の使用法については，以前のプリント，もしくはヘルプを参照すること。

これらの課題に対して，次のような出力が得られるように LaTeX で記述し，PDF ファイルを作成して CHORUS で提出せよ。ただし，枠で囲まれた部分については，該当する Mathematica の計算結果を記入すること。

計算機１・２ 第11回 課題

学籍番号: ********　　氏名: ********

平成19年7月10日

課題1　ロー法の基礎

(1) Mathematica での関数 `P[m,k]` の定義を記す

(2)

m	最小の k	$k/\sqrt{m}$
1,000,000	(ここを計算する)	(ここを計算する)
10,000,000	(ここを計算する)	(ここを計算する)
100,000,000	(ここを計算する)	(ここを計算する)

計算の結果より，50%を越す最小の k は (定数)$\times\sqrt{m}$ 程度であることが分かる。

課題2　試し割り法とロー法の実行時間の比較

m	m の約数	試し割り法	ロー法
17030036839	(ここを計算する)	(ここを計算する)	(ここを計算する)
63491454743	(ここを計算する)	(ここを計算する)	(ここを計算する)