再帰関数Ⅰ

０．目次

１．階乗関数

再帰関数は、関数の中で自分自身を呼び出す関数をいう。関数を簡潔に定義することができる。
階乗関数f(n) (n≧0)を明示的に書くとつぎのようになる。

```
f(n) = n*(n-1)*(n-2)・・・2*1  (n≧1)
     = 1                      (n=0 )
```

●再帰的定義

階乗f(n)(n≧0)は、つぎのように再帰的に定義される。

```
f(n) = n * f(n-1)   (n≧1)
     = 1            (n=0 )
```

●階乗関数f(n)の計算。n=4の場合。

```
f(4) = 4*f(3)
     = 4*3*f(2)
     = 4*3*2*f(1)
     = 4*3*2*1
     = 24
```

階乗関数f(n)は、定義通りに書ける。

●プログラム(rf111.c)

```
1  /* << rf111.c >> */
2  #include <stdio.h>
3  int f(int n);
4
5  int main() {
6    int n; /* 正整数n。*/
7
8    /* 正整数nの読み込み。*/
9    scanf("%d",&n);
10
11   /* 階乗関数f(n)の計算。*/
12   printf("%d ! = %d ¥n",n,f(n));
13 }
14
15 /* 階乗関数。*/
16 int f(int n) {
17   int z;
18   if( n == 0 ) {
19     z = 1;
20   } else {
21     z = n*f(n-1);
22   }
23   return z;
24 }
```

実行結果

```
% cc rf111.c
% ./a.out
4
4 ! = 24
```

●n=4 の場合、プログラムrf111.cの動作は①から⑨のようになる。

⑨f(4)は値24を返す。

①f(4)はf(3)を呼び出す。 f(4) ⇅ f(3) ⑧f(3)は値6をf(4)に返す。

②f(3)はf(2)を呼び出す。 f(3) ⇅ f(2) ⑦f(2)は値2をf(3)に返す。

③f(2)はf(1)を呼び出す。 f(2) ⇅ f(1) ⑥f(1)は値1をf(2)に返す。

④f(1)はf(0)を呼び出す。 f(1) ⇅ f(0) ⑤f(0)は値1をf(1)に返す。

●プログラム(rf111.c)において、関数fが呼び出される回数

n	呼び出される回数
4	5
5	6
6	7
7	8

● 階乗関数f(n)の動作解析

プログラム(rf112.c)

```
 1 /* << rf112.c >> */
 2 #include <stdio.h>
 3 int f(int n);
 4
 5 int main() {
 6   int n;
 7   n = 4;
 8   printf("%d ! = %d ¥n",n,f(n));
 9 }
10 int f(int n) {
11   int z;
12   printf("--> n=%d ¥n",n);
13   if( n == 0 ) {
14     z = 1;
15   } else {
16     z = n * f(n-1);
17   }
18   printf("<-- n=%d z=%d ¥n",n,z);
19   return z;
20 }
```

実行結果

```
 1 % cc rf112.c
 2 % a.out
 3 --> n=4
 4 --> n=3
 5 --> n=2
 6 --> n=1
 7 --> n=0
 8 <-- n=0 z=1
 9 <-- n=1 z=1
10 <-- n=2 z=2
11 <-- n=3 z=6
12 <-- n=4 z=24
13 4 ! = 24
```

２．基本演算

２．１　乗算

●再帰的定義

m,nを正整数とする。

```
mul(n,m) = 0                (m=0)
         = n + mul(n,m-1)   (m≧1)
```

●乗算関数mul(n,m)の計算。n=5,m=3の場合。

```
mul(5,3) = 5 + mul(5,2)
         = 5 + 5 + mul(5,1)
         = 5 + 5 + 5 + mul(5,0)
         = 5 + 5 + 5 + 0
         = 15
```

乗算関数mul(n,m)は、定義通りに書ける。

●プログラム(rf211.c)

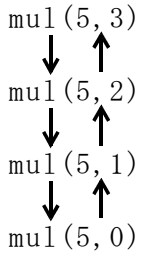
```
 1  /* << rf211.c >> */
 2  #include <stdio.h>
 3  int mul(int n, int m);
 4
 5  int main() {
 6    int m,n;
 7
 8    while( 1 ) {
 9      /* 正整数n,mの読み込み。*/
10      scanf("%d%d",&n,&m);
11      if( (n <= 0)||(m <= 0) ) { break; }
12
13      printf("%d * %d = %d ¥n",n,m,mul(n,m));
14    }
15  }
16
17  /* 乗算関数。*/
18  int mul(int n, int m) {
19    if( m == 0 ) {
20      return 0;
21    } else {
22      return n+mul(n,m-1);
23    }
24  }
```

実行結果

```
% cc rf211.c
% ./a.out
2 3
2 * 3 = 6
-1 -1
```

●プログラムrf211.cの動作。n=5,m=3の場合、①から⑦のようになる。

① mul(5,3)がmul(5,2)を呼び出す。
② mul(5,2)がmul(5,1)を呼び出す。
③ mul(5,1)がmul(5,0)を呼び出す。



④ mul(5,0)が0を返す。
⑤ mul(5,1)が5を返す。
⑥ mul(5,2)が10を返す。
⑦ mul(5,3)が15を返す。

●プログラム(rf211.c)において、n,mが与えられたとき、関数mulが呼び出される回数

n	m	呼び出される回数
123	45	46
45	123	124
2468	1357	1358
1948	1007	1008

２．２　除算

●再帰的定義

m,nを正整数とする。nをmで割る。

```
div(n,m) = 0                (n<m)
         = 1 + div(n-m,m)   (n≧m)
```

●除算関数div(n,m)の計算。n=15,m=4の場合。

```
div(15,4) = 1 + div(11,4)
          = 1 + 1 + div(7,4)
          = 1 + 1 + 1 + div(3,4)
          = 3
```

除算関数div(n,m)を求めるプログラムはつぎのようになる。

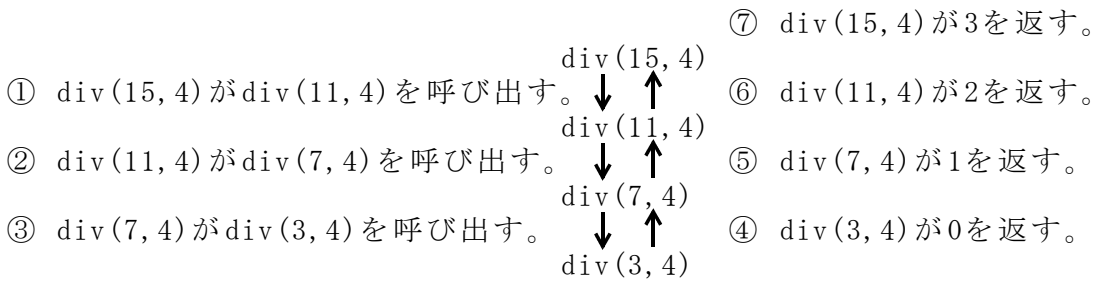
●プログラム(rf221.c)

```
 1  /* << rf221.c >> */
 2  #include <stdio.h>
 3  int div(int n, int m);
 4
 5  int main() {
 6    int m,n;
 7
 8    while( 1 ) {
 9      /* 正整数n,mの読み込み。*/
10      scanf("%d%d",&n,&m);
11      if( (n <= 0)||(m <= 0) ) { break; }
12
13      printf("%d / %d = %d ¥n",n,m,div(n,m));
14    }
15  }
16
17  /* 除算関数。*/
18  int div(int n, int m) {
19    if( n < m ) {
20      return 0;
21    } else {
22      return 1+div(n-m,m);
23    }
24  }
```

実行結果

```
% cc rf221.c
% ./a.out
12 3
12 / 3 = 4
12 4
12 / 4 = 3
12 5
12 / 5 = 2
-1 -1
```

●プログラムrf221.cの動作。n=15,m=4の場合、①から⑦のようになる。



●プログラム(rf221.c)において、n,mが与えられたとき、関数divが呼び出される回数

n	m	呼び出される回数
123	45	3
45	123	1
2468	13	190
1948	10	195

２．３　剰余

m,nを正整数とする。nをmで割ったとき得られる剰余mod(n,m)の再帰的定義はつぎのようになる。

$$\begin{aligned} mod(n,m) &= n && (n<m) \\ &= mod(n-m,m) && (n \geqq m) \end{aligned}$$

剰余関数mod(n,m)を求めるプログラムはつぎのようになる。

●プログラム(rf231.c)

```
1  /* << rf231.c >> */
2  #include <stdio.h>
3  int mod(int n, int m);
4
5  int main() {
6    int m,n;
7
8    while( 1 ) {
9      /* 正整数n,mの読み込み。*/
10     scanf("%d%d",&n,&m);
11     if( (n<=0)||(m<=0) ) { break; }
12
13     printf("mod(%d,%d) = %d ¥n",n,m,mod(n,m));
14   }
15 }
16
17 /* 剰余関数。*/
18 int mod(int n, int m) {
19   if( n < m ) {
20     return n;
21   } else {
22     return mod(n-m,m);
23   }
24 }
```

実行結果

```
% cc rf231.c
% ./a.out
15 3
mod(15,3) = 0
15 4
mod(15,4) = 3
-1 -1
```

●プログラム(rf231.c)において、n,mが与えられたとき、関数modが呼び出される回数

n	m	呼び出される回数
123	45	3
45	123	1
2468	13	190
1948	10	195

３．最大公約数

正整数a,bの最大公約数をユークリッド互除法で求める。

a = bq + r (0＜r＜b) のとき、aとbの最大公約数gcd(a,b)とbとrの最大公約数gcd(b,r)が一致することに基づく。すなわち、gcd(a,b)=gcd(b,r)。
a=155, b=40 の場合,

```
155 = 40 * 3 + 35      gcd(155,40)=gcd(40,35)
 40 = 35 * 1 + 5       gcd(40,35)=gcd(35,5)
 35 = 5 * 7            gcd(35,5)=5
```

となる。

（考察） gcd(a,b)=gcd(b,r)の証明。

a = bq + r (0<r<b) のとき、a,bの公約数の集合をG(a,b)とおくと、

G(a,b) = G(b,r)

が成り立つ。これは、a,bの最大公約数とb,rの最大公約数が一致することを意味する。

G(a,b)⊂G(b,r)を示す。

a=a'k, b=b'k とすると、a'k=b'kq+r となり、変形して、r=(a'-b'q)k となる。これは、b,rが約数kを持つことを意味する。
すなわち、G(a,b)⊂G(b,r)が示された。

G(a,b)⊃G(b,r)を示す。

b=b'm, r=r'm とすると、a=b'mq+r'm となり、変形して、a=(b'q+r')m となる。これは、a,bが約数mを持つことを意味する。
すなわち、G(a,b)⊃G(b,r)が示された。

以上より、G(a,b) = G(b,r) が証明された。

●再帰的定義

```
gcd(a,b) = gcd(b,r)   (a=bq+r,r>0)
         = b          (a=bq+r,r=0)
```

関数gcd(a,b)は、定義通りに書ける。

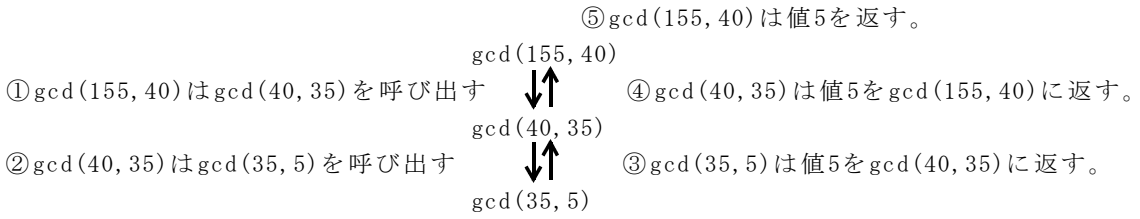
●プログラム(rf311.c)

```
1  /* << rf311.c >> */
2  #include <stdio.h>
3  int gcd(int a, int b);
4
5  int main() {
6    int a,b;
7
8    while( 1 ) {
9      /* 正整数a,bの読み込み。*/
10     scanf("%d%d",&a,&b);
11     if( (a <= 0)||(b <= 0) ) { break; }
12
13     printf("gcd(%d,%d) = %d ¥n",a,b,gcd(a,b));
14   }
15 }
16
17 /* 最大公約数。*/
18 int gcd(int a, int b) {
19   int r,z;
20   r = a % b;
21   if( r == 0 ) {
22     z = b;
23   } else {
24     z = gcd(b,r);
25   }
26   return z;
27 }
```

実行結果

```
% cc rf311.c
% ./a.out
1234 567
gcd(1234,567) = 1
120 64
gcd(120,64) = 8
0 0
```

●a=155,b=40の場合、プログラムrf311.cの動作は①から⑤のようになる。



●プログラム(rf311.c)において、a,bが与えられたとき、関数gcdが呼び出される回数

a	b	呼び出される回数
123	45	5
45	123	6
120	64	2
89	55	9

●関数gcd(a,b)の動作解析

プログラム(rf312.c)

```
 1  /* << rf312.c >> */
 2  #include <stdio.h>
 3  int gcd(int a, int b);
 4
 5  int main() {
 6    int a,b;
 7    a = 123; b = 45;
 8    printf("gcd(%d,%d) = %d ¥n",a,b,gcd(a,b));
 9  }
10  int gcd(int a, int b) {
11    int r,z;
12    printf("--> a=%d b=%d ¥n",a,b);
13    r = a % b;
14    if( r == 0 ) {
15      z = b;
16    } else {
17      z = gcd(b,r);
18    }
19    printf("<-- a=%d b=%d z=%d ¥n",a,b,z);
20    return z;
21  }
```

実行結果

```
 1  % cc rf312.c
 2  % ./a.out
 3  --> a=123 b=45
 4  --> a=45 b=33
 5  --> a=33 b=12
 6  --> a=12 b=9
 7  --> a=9 b=3
 8  <-- a=9 b=3 z=3
 9  <-- a=12 b=9 z=3
10  <-- a=33 b=12 z=3
11  <-- a=45 b=33 z=3
12  <-- a=123 b=45 z=3
13  gcd(123,45) = 3
```

４．フィボナッチ関数

●再帰的定義

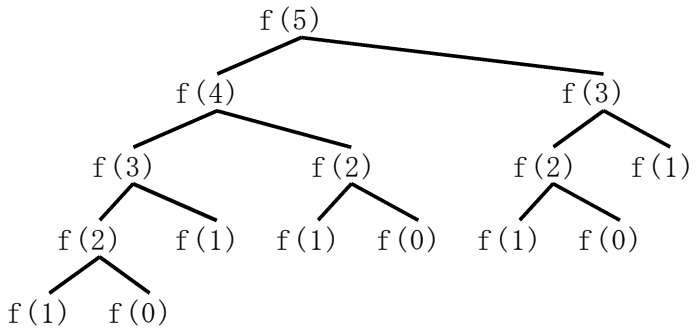
フィボナッチ関数はつぎのように定義される。

```
f(n) = f(n-1) + f(n-2)   (n≧2)
     = 1                 (n=1 )
     = 1                 (n=0 )
```

●フィボナッチ関数f(n)の計算。n=5 の場合。

```
f(5) = f(4) + f(3)
     = f(3) + f(2) + f(2) + f(1)
     = f(3) + 2*f(2) + f(1)
     = f(2) + f(1) + 2*(f(1) + f(0)) + f(1)
     = f(2) + 4*f(1) + 2*f(0)
     = f(1) + f(0) + 4*f(1) + 2*f(0)
     = 5*f(1) + 3*f(0)
     = 8
```

●フィボナッチ関数f(n)の計算状況。n=5 の場合。



（注意）同じ値f(k)が
何度も計算され、
効率が悪くなる。

f(5)	1 回
f(4)	1 回
f(3)	2 回
f(2)	3 回
f(1)	5 回
f(0)	3 回

フィボナッチ関数f(n)は、定義通りに書ける。

●プログラム(rf411.c)

```
 1  /* << rf411.c >> */
 2  #include <stdio.h>
 3  int f(int n);
 4
 5  int main() {
 6    int n;
 7
 8    /* 正整数nの読み込み。*/
 9    scanf("%d",&n);
10    printf("f(%d) = %d ¥n",n,f(n));
11  }
12
13  /* フィボナッチ関数。*/
14  int f(int n) {
15    int z;
16    if( (n==0)||(n==1) ) {
17      z = 1;
18    } else {
19      z = f(n-1) + f(n-2);
20    }
21    return z;
22  }
```

実行結果

```
% cc rf411.c
% ./a.out
9
f(9) = 55
```

●プログラム(rf411.c)において、関数fが呼び出される回数

n	呼び出される回数
3	5
4	9
5	15
6	25

●フィボナッチ関数f(n)の動作解析

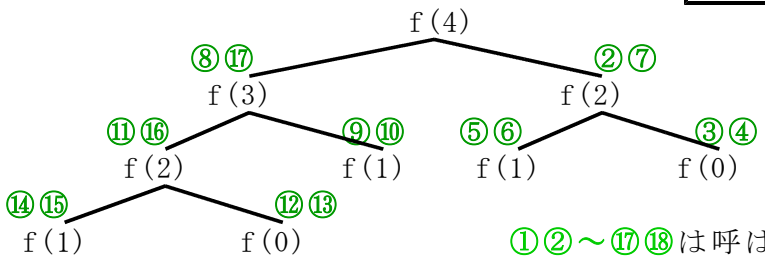
プログラム(rf412.c)

```
 1  /* << rf412.c >> */
 2  #include <stdio.h>
 3  int f(int n);
 4
 5  int main() {
 6    int n;
 7    n = 4;
 8    printf("f(%d) = %d ¥n",n,f(n));
 9  }
10  int f(int n) {
11    int z;
12    printf("--> n=%d ¥n",n);
13    if( (n==0)||(n==1) ) {
14      z = 1;
15    } else {
16      z = f(n-1) + f(n-2);
17    }
18    printf("<-- n=%d z=%d ¥n",n,z);
19    return z;
20  }
```

実行結果

```
% cc rf412.c
% a.out
①  --> n=4
②  --> n=2
③  --> n=0
④  <-- n=0 z=1
⑤  --> n=1
⑥  <-- n=1 z=1
⑦  <-- n=2 z=2
⑧  --> n=3
⑨  --> n=1
⑩  <-- n=1 z=1
⑪  --> n=2
⑫  --> n=0
⑬  <-- n=0 z=1
⑭  --> n=1
⑮  <-- n=1 z=1
⑯  <-- n=2 z=2
⑰  <-- n=3 z=3
⑱  <-- n=4 z=5
f(4) = 5
```

関数f(n)が呼び出される順序　①⑱



①②～⑰⑱は呼ばれる順序

大きいnの値について実行時間を測定する。

●プログラム(rf413.c)

```
 1  /* << rf413.c >> */
 2  #include <stdio.h>
 3  double f(int n);
 4
 5  int main() {
 6    int n;
 7
 8    /* 正整数nの読み込み。*/
 9    scanf("%d",&n);
10    printf("f(%d) = %16.0f¥n",n,f(n));
11  }
```

```
12
13 /* フィボナッチ関数。*/
14 double f(int n) {
15   double z;
16   if( (n==0)||(n==1) ) {
17     z = 1.0;
18   } else {
19     z = f(n-1) + f(n-2);
20   }
21   return z;
22 }
```

実行時間

```
% cc rf413.c
% time ./a.out
45
f(45) =        1836311903
16.927u 0.007s 0:18.23 92.8%    0+0k 0+0io 0pf+0w
% time ./a.out
46
f(46) =        2971215073
27.398u 0.008s 0:29.36 93.2%    0+0k 0+8io 0pf+0w
% time ./a.out
47
f(47) =        4807526976
44.325u 0.029s 0:49.87 88.9%    0+0k 0+0io 0pf+0w
```

指定した範囲のf(n)(0≦n≦10) について、計算結果を配列に保存し、２度目以降計算をしないようにして、プログラム（rf413.c）を改良する。

●プログラム(rf413a.c)

```
1  /* << rf413a.c >> */
2  #include <stdio.h>
3  #define N 10 /* 事前に計算しておく最大値。*/
4  double g[N+1]; /* 保存用の配列。*/
5  double f(int n);
6
7  int main() {
8    int i,n;
9
10   /* 正整数nの読み込み。*/
11   scanf("%d",&n);
12
13   /* 初期設定。*/
14   g[0] = 1; g[1] = 1;
15   for( i=2; i<=N; i++ ) {
16     g[i] = g[i-1] + g[i-2];
17   }
```

```
18
19    printf("f(%d) = %16.0f¥n",n,f(n));
20  }
21
22  /* フィボナッチ関数。*/
23  double f(int n) {
24    double z;
25    if( n <= N ) {
26      z = g[n];
27    } else {
28      z = f(n-1) + f(n-2);
29    }
30    return z;
31  }
```

実行時間

```
% cc rf413a.c
% time ./a.out
45
f(45) =        1836311903
0.255u 0.000s 0:01.49 16.7%     0+0k 0+0io 0pf+0w
% time ./a.out
46
f(46) =        2971215073
0.417u 0.000s 0:01.62 25.3%     0+0k 0+0io 0pf+0w
% time ./a.out
47
f(47) =        4807526976
0.666u 0.000s 0:02.74 24.0%     0+0k 0+0io 0pf+0w
```

５．べき乗関数

べき乗（a^n、a：実数、n：正整数）の計算法を考察する。

５．１　解法１

●再帰的定義１

$$a^n = a^{n-1} * a \quad (n \geqq 1)$$
$$= 1 \quad (n=0)$$

再帰的定義１で、べき乗関数g1(a,n)を求める。

●プログラム(rf511.c)

```
1  /* << rf511.c >> */
2  #include <stdio.h>
3  float g1(float a, int n);
4
5  int main() {
6    int n;
7    float a;
8
9    while( 1 ) {
10     /* 実数aと正整数nの読み込み。*/
11     scanf("%f%d",&a,&n);
12     if( n < 0 ) { break; }
13
14     printf("%f ** %d = %f ¥n",a,n,g1(a,n));
15   }
16 }
17
18 /* べき乗の計算。*/
19 float g1(float a, int n) {
20   if( n == 0 ) {
21     return 1.0;
22   } else {
23     return a*g1(a,n-1);
24   }
25 }
```

実行結果

```
% cc rf511.c
% ./a.out
2.0 2
2.000000 ** 2 = 4.000000
2.0 20
2.000000 ** 20 = 1048576.000000
0.0 -1
```

●プログラム(rf511.c)において、a,nが与えられたとき、関数g1が呼び出される回数

a	n	呼び出される回数
2	5	6
2	8	9
2	12	13
2	15	16

５．２　解法２

●再帰的定義２

$$
\begin{aligned}
a^n &= a^{n/2} * a^{n/2} && \text{nが偶数の場合} \\
&= a^{n-1} * a && \text{nが奇数の場合} \\
&= 1 && \text{n=0の場合}
\end{aligned}
$$

●計算例

$$
\begin{aligned}
a^{20} &= a^{10} \times a^{10} \\
a^{10} &= a^{5} \times a^{5} \\
a^{5} &= a^{4} \times a \\
a^{4} &= a^{2} \times a^{2} \\
a^{2} &= a \times a
\end{aligned}
$$

再帰的定義２で、べき乗関数g2(a,n)を求める。

●プログラム(rf521.c)

```
 1 /* << rf521.c >> */
 2 #include <stdio.h>
 3 float g2(float a, int n);
 4
 5 int main() {
 6   int n;
 7   float a;
 8
 9   while( 1 ) {
10     /* 実数aと正整数nの読み込み。*/
```

```
11      scanf("%f%d",&a,&n);
12      if( n < 0 ) { break; }
13
14      printf("%f ** %d = %f ¥n",a,n,g2(a,n));
15    }
16  }
17
18  /* べき乗の計算。*/
19  float g2(float a, int n) {
20    float b;
21    if( n == 0 ) { return 1.0; }
22    if( n%2 == 0 ) {
23      b = g2(a,n/2); return b*b;
24    } else {
25      b = g2(a,n-1); return b*a;
26    }
27  }
```

実行結果

```
% cc rf521.c
% ./a.out
2.0 2
2.000000 ** 2 = 4.000000
2.0 20
2.000000 ** 20 = 1048576.000000
0.0 -1
```

●プログラム(rf521.c)において、a,nが与えられたとき、関数g2が呼び出される回数

a	n	呼び出される回数
2	5	5
2	8	5
2	12	6
2	15	8