階乗 Factorial の計算

10! すなわち10の階乗を計算すると3,628,800 と大きな数になる。
5!以上の階乗は、素因数 2^d 3^e 5^f からして10の倍数になり、下の桁には必ず 0が並ぶ。

普通の関数電卓での計算では、17! = 355,687,428,096,000 あたりが限界だ。
これ以上大きいと指数表示になり、下の桁の数字か分からない。

エクセルで20!まで計算できた結果を下に示す。
100! さらに 1000! は、もちろん、大きすぎて計算できていない。
せめて下の桁に0がいくつ並ぶかわかる方法はないものだろうか。

Cプログラミングで階乗関数 Factorial function を作ってみた。
このプログラミングでも100!を求めることはできない。

n	n!
1	1
2	2
3	6
4	24
5	120
6	720
7	5,040
8	40,320
9	362,880
10	3,628,800
11	39,916,800
12	479,001,600
13	6,227,020,800
14	87,178,291,200
15	1,307,674,368,000
16	20,922,789,888,000
17	355,687,428,096,000
18	6,402,373,705,728,000
19	121,645,100,408,832,000
20	2,432,902,008,176,640,000
25	1.55E+25
50	3.04E+64
65	8.25E+90
100	9.33E+157
1000	#NUM!
10000	#NUM!

```
/* 階乗の計算 n!  S. Kusafusa */
/*  最大 何桁まで計算できる */
#include <stdio.h>
int main()
{
    int n, i;
    unsigned long long factorial = 1;

    scanf("%d",&n);
    printf("Enter an integer:%d\n", n);

    if (n < 0)
        printf("Error!");

    else
    {
        for(i=1; i<=n; ++i)
        {
            factorial *= i;
        }
        printf("%d!= %llu", n, factorial);
    }
    return 0;
}
```

計算値:

```
18! = Enter an integer:18
18! = 6402373705728000


Enter an integer:19
19! = 121645100408832000

Enter an integer:20
20! = 2432902008176640000
```

では、20以上の計算では、直接に計算しなくても、せめて、下の桁に並ぶ0の桁数だけでも計算することはできないだろうか。素数の素因数分解から、探ってみる。

n! の場合、

5^f に着目して、f(n) とおくと、f(1) = 0、f(5) = 1、f(10) =2、f(100) = 24、. . 、100!の場合24個の0が並ぶ。

n!を素因数分解するとき、素数pの指数を f(n)とおいて、f(n) = f(n1) + n1 と書ける。

10^r!の場合、pを5とすると、

f(1000) = f(200) + 200
f(200) = f(40) + 40
f(40) f(8) + 8
f(8) = 1
∵ f(1000) = 200 + 40 + 8 +1 = 249

f(10000) = f(2000) + 2000
f(2000) = f(400) + 400
f(400) = f(80) + 80
f(80) = f(16) + 16
f(16) = f(3) + 3
∵ f(10000) = 2000 + 400 + 80 + 16 + 3 = 2499

n	10^n	10^n!の下の桁の0の個数
1	10	2
2	100	24
3	1000	249
4	10000	2499
5	100000	24999
6	1000000	249998
7	10000000	2499999
8	100000000	24999999
9	1000000000	249999998
10	10000000000	2499999997

n	10^n	10^n!の下の桁の0の個数																
1	10	2	2	0														2
2	100	24	20	4														24
3	1000	249	200	40	8	1												249
4	10000	2499	2000	400	80	16	3											2499
5	100000	24999	20000	4000	800	160	32	6	1									24999
6	1000000	249998	200000	40000	8000	1600	320	64	13	2								249998.8
7	10000000	2499999	2000000	400000	80000	16000	3200	640	128	25	5							2499998
8	100000000	24999999	20000000	4000000	800000	160000	32000	6400	1280	256	51	10	2					24999999
9	1000000000	249999998	200000000	40000000	8000000	1600000	320000	64000	12800	2560	512	102	20	4				249999998
10	10000000000	2499999997	2000000000	400000000	80000000	16000000	3200000	640000	128000	25600	5120	1024	204	40	8	1		2499999997

次に、p^n! のpの指数を考える。

p = 2 の場合

n	2^n ベキ乗	階乗	f(2^n)	指数
1	2	2	1	1
2	4	24	2	3
3	8	40,320	4	7
4	16	20,922,789,888,000	8	15
5	32	################################	16	31
6	64	################################	32	63
7	128	################################	64	127
8	256	#NUM!	128	255
9	512	#NUM!	256	511
10	1024	#NUM!	512	1023

p = 5の場合

n	5^n ベキ乗	階乗	f(5^n)	指数
1	5	120	1	1
2	25	15,511,210,043,331,000,000,000,000	5	6
3	125	##############################	25	31
4	625	#NUM!	125	156
5	3125	#NUM!	625	781
6	15625	#NUM!	3125	3906
7	78125	#NUM!	15625	19531
8	390625	#NUM!	78125	97656
9	1953125	#NUM!	390625	488281
10	9765625	#NUM!	1953125	2441406

###. . . ##は、桁数オーバー、　#NUM!は、計算のオーバーフロー・エラーで計算できず

課題：　P=5の指数と、メルセンヌ数へのアナロジーを考えよ

Rubyで階乗を計算すると、簡単！

```
# 65の階乗を求める
ans = 1
for i in 1..65
  ans *= i
end
# 出力する
print "65! = ", ans, "\n"
```

計算結果:

```
65! =
8247650592082470666723170306785496252186258551345437492922123134388955774976
000000000000000
```

gmp.hなしのCプログラミングではできなくとも、多倍長を持つRubyにはできる

```
# 100の階乗を求める
ans = 1
for i in 1..100
  ans *= i
end
# 出力する
print "100! = ", ans, "\n"
```

計算結果:

```
100! =
9332621544394415268169923885626670049071596826438162146859296389521759999322
9915608941463976156518286253697920827223758251185210916864000000000000000000
000000
```

written by C. Кусафуса　草房誠二郎　S. Kusafusa（くさふさせいじろう)

Dec 2016