配列の応用：大きな整数の計算

[例題] 次のような階乗の表を作ってみよう。

```
 2! =                                                        2
 3! =                                                        6
 4! =                                                       24
 5! =                                                      120
 6! =                                                      720
 7! =                                                     5040
 8! =                                                    40320
 9! =                                                   362880
10! =                                                  3628800

11! =                                                 39916800
12! =                                                479001600
13! =                                               6227020800
14! =                                             8 7178291200
15! =                                           130 7674368000
16! =                                          2092 2789888000
17! =                                         35568 7428096000
18! =                                        640237 3705728000
19! =                                      12164510 0408832000
20! =                                     243290200 8176640000

21! =                                    5109094217 1709440000
22! =                                 11 2400072777 7607680000
23! =                                258 5201673888 4976640000
24! =                               6204 4840173323 9439360000
25! =                             155112 1004333098 5984000000
26! =                            4032914 6112660563 5584000000
27! =                          108888694 5041835216 0768000000
28! =                         3048883446 1171386050 1504000000
29! =                       8 8417619937 3970195454 3616000000
30! =                     265 2528598121 9105863630 8480000000
```

[練習] for 文を用いて素直に階乗を計算するプログラムを作りなさい。

関数の再帰的定義

階乗の計算は次のように再帰的に定義できる。

$$\mathrm{fact(n)}=\begin{cases} 1, & (\mathrm{n}\text{ が }1\text{ の時}) \\ \mathrm{n}\times\mathrm{fact(n-1)}, & (\mathrm{n}\text{ が }1\text{ より大きい時}) \end{cases}$$

C 言語は、このような再帰的定義をそのままプログラムにすることができる。

```
 1  #include <stdio.h>
 2  int fact(int);
 3
 4  main() {
 5    int i;
 6
 7    for(i=1; i<=10; i++)                    /* 10 までの階乗を順に求める */
 8      printf("%2d! = %20d\n", i, fact(i));  /* 20 桁分のスペースに右詰め表示 */
 9  }
10
11  int fact(int i) {                         /* 階乗の再帰的定義そのまま */
12    if (i==1) return 1;
13    else return(i*fact(i-1));
14  }
```

ただ、このようにしても long 型の変数が取り得る上限以上の数は扱うことができないので、もっと別の工夫が必要となる。

大きな整数を扱うには…

階乗の値を格納する変数を int(または long) で宣言してもその上限は 2147483647 であるから、30!は求まらない。では、double で宣言すればよいかというと上限はカバーできるかもしれないが、有効数字が上 15 桁しかないので、正確な値は求まらない。では、どうしたらよいだろうか?
まず、大きな整数をどのようにコンピュータの中で格納するか考えよう。int, long, float, double といった C 言語で提供されている変数の型を単に使うだけでは、大きな整数は表せない。そこで、各桁の数を 1 つの整数変数に対応させる。整数変数としては i, j, k,…を使っても良いがここでは配列を使い、1 の位は a[0], 10 の位は a[1], 100 の位は a[2], 1000 の位は a[3], …に格納することにする。

今、15 桁の数を取り扱うことにすると、整数 1 は、

a[14]	a[13]	a[12]	a[11]	a[10]	a[9]	a[8]	a[7]	a[6]	a[5]	a[4]	a[3]	a[2]	a[1]	a[0]
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

となっているはずである。この整数に 2 を掛けて 2 の階乗を求めるには、a[0] から a[14] までのすべての桁に 2 を掛ければよい。結果として

a[14]	a[13]	a[12]	a[11]	a[10]	a[9]	a[8]	a[7]	a[6]	a[5]	a[4]	a[3]	a[2]	a[1]	a[0]
0	0	0	0	0	0	0	0	0	0	0	0	0	0	2

となる。この数に 3 を掛けて 3 の階乗を求めるには、同様に a[0] から a[14] までのすべての桁に 3 を掛けて

a[14]	a[13]	a[12]	a[11]	a[10]	a[9]	a[8]	a[7]	a[6]	a[5]	a[4]	a[3]	a[2]	a[1]	a[0]
0	0	0	0	0	0	0	0	0	0	0	0	0	0	6

となる。この数に 4 を掛けて 4 の階乗を求めるには、同様に a[0] から a[14] までのすべての桁に 4 を掛ければよいのだが、a[0] に 4 を乗ずると 24 になる。しかし、a[0] は 1 の位の数だから、0 から 9 までの整数しかとってはいけない。すなわち 24 の 2 は繰り上がって次の桁すなわち a[1] に足されなければならない。結果として

a[14]	a[13]	a[12]	a[11]	a[10]	a[9]	a[8]	a[7]	a[6]	a[5]	a[4]	a[3]	a[2]	a[1]	a[0]
0	0	0	0	0	0	0	0	0	0	0	0	0	2	4

となる。この整数に 5 を掛けて 5 の階乗を求めるには、各位に 5 を乗じてから繰り上がりの分を隣の桁に足して

a[14]	a[13]	a[12]	a[11]	a[10]	a[9]	a[8]	a[7]	a[6]	a[5]	a[4]	a[3]	a[2]	a[1]	a[0]
0	0	0	0	0	0	0	0	0	0	0	0	1	2	0

となる。繰り上がりの処理は下の桁から行わなければいけないことに注意しよう。

このような作業で、配列要素数分の桁の整数が計算できる。

```
 1  /* big factorial number */
 2  #include <stdio.h>
 3  #define N 60                /* 桁 */
 4  #define FACT 30             /* 階乗上限 */
 5  main ()
 6  {
 7    int a[N];
 8    int i,n,carry;
 9
10    a[0]=1;                   /* 初期値は 000.....001 */
11    for (i=1; i<N; i++)
12          ┌──────────┐
13          │          │;
14          └──────────┘
15
16    for(n=2; n<=FACT; n++) {        /* 階乗を順に計算する */
17      carry=0;
18      for(i=0; i<N; i++) {
19        a[i]=a[i]*n;          /* 各桁に数を乗じて繰り上がりを隣の桁に足し込む */
20        a[i]=a[i]+carry;      /* carry は繰り上がり分 */
21        carry = a[i]/10;
22              ┌──────┐
23        a[i]= │      │;
24              └──────┘
25      }
26
27      printf("%2d! = ",n);          /* 結果の出力 */
28      for(i=N-1; i>=0; i--)         /* a[N-1] から a[0] までを順に出力する */
29          ┌──────────┐
30          │          │;
31          └──────────┘
32      printf("\n");
33    }
34  }
```

[練習] 上のプログラムの実行結果では見やすくないので、

- 頭の不要な 0 は出力しない
- 10 桁ごとにスペース文字を入れる

等の工夫をしなさい。