整数型についての補足

- 整数型 int が扱うことができる整数の範囲は処理系に依存
- limits.h で INT_MAX, INT_MIN 等が定義されている
- 今時（2026年）の多くの処理系では int は4バイト = 32ビットで表す
- 2^32 = 4,294,967,296 (42億・・・)、正負を含めると ± 21億・・・
- 符号無し整数の型として unsigned int がある
- 更に大きな整数を扱うために long int と unsigned long int がある
- 8バイト = 64ビット、2^64 = 18,446,744,073,709,551,616

大きな（桁数が多い）整数の取り扱い

int, unsigned int, long int, unsigned long int について確認

```
#include <stdio.h>
#include <limits.h>

int main(void)
{
  printf(“ INT_MAX = %d\n", INT_MAX);
  printf(“ INT_MIN = %d\n", INT_MIN);
  printf("UINT_MAX = %u\n", UINT_MAX);

  printf(“ LONG_MAX = %ld\n", LONG_MAX);
  printf(“ LONG_MIN = %ld\n", LONG_MIN);
  printf("ULONG_MAX = %lu\n", ULONG_MAX);

  return 0;
}
```

変換指定に注意:
int —> %d, %u
long int —> %ld, %lu を使う

大きな数の素因数分解

素因数分解を行うプログラムを用いて巨大数の素因数分解は時間がかかることを確認せよ

```
#include <stdio.h>
int main(void)
{
  unsigned long a, pm1, pm2, pm3, pm4, pm5;

  // 大きな素数 wikipediaから抜粋
  pm1 = 131071;
  pm2 = 524287;
  pm3 = 6700417;
  pm4 = 2147483647;
  pm5 = 67280421310721;

  a = pm3;
  printf(“%lu = ", a);

  // 試し割りアルゴリズム：割り切れるかどうか試すやり方
  i=2;

  do{
    while( a%i == 0 ){
      a = a/i;
      printf("%lu ", i);
    }
    i++;
  } while( a > 1 );

  printf("\n");

  return 0;
}
```

桁数が大きい掛け算は容易だが、
桁数が大きな数の素因数分解は極めて
長時間の計算が必要

デジタル署名などで使用されるRSA暗
号の根拠となっている