問題集（アシアルテキスト対応）VI 繰り返しと条件分岐の組み合わせ

繰り返し（反復）と、選択（分岐）を組み合わせると、複雑なプログラムを作ることができます。

これまで、様々な問題を解くために、その手順（アルゴリズム）が考えられてきましたが、多くのアルゴリズムは繰り返し（反復）と選択（分岐）の組み合わせで出来ています（※もちろん、繰り返しも条件分岐も使わない問題・アルゴリズムもあります)。

たとえば、「２つの整数の最大公約数を求める」という問題を解く手順の１つに、『ユークリッドの互除法』があります。この手順も、繰り返し（反復）と選択（分岐）の組み合わせで出来ています。

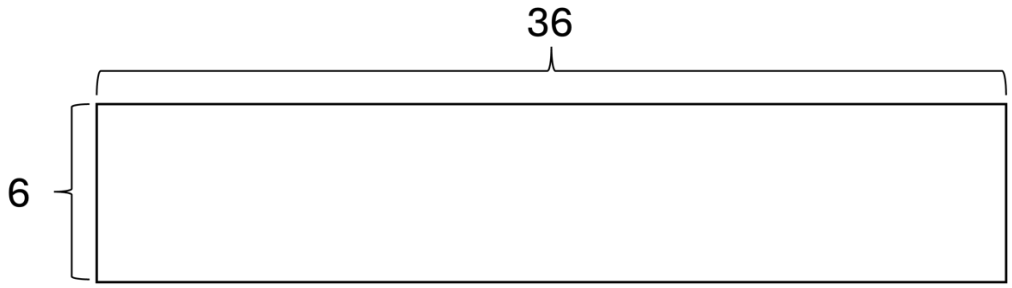
この問題では、「ユークリッドの互除法」を実現するプログラムを作成しますが、プログラムを書く前に、図も使いながら、その考え方を確認します。

ユークリッドの互除法の基本の考え方は、およそ次の通りです。

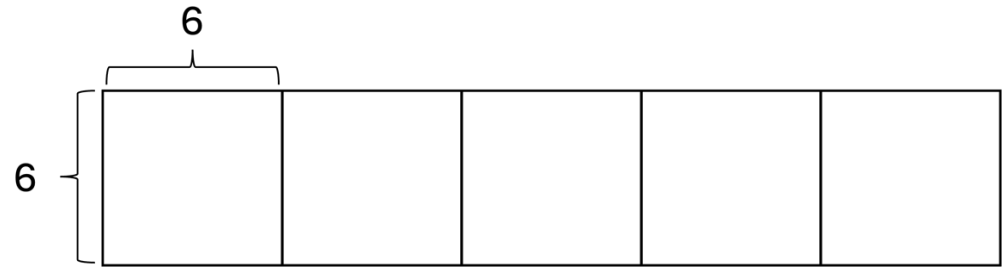
1. ２つの値を、それぞれ長方形の辺の長さと見なす
2. 1.で得られた長方形を、「短辺の長さを辺の長さにする正方形」に分ける。具体的には、長辺の長さを短辺の長さで割り、余りを求める
   (ア) 余らずに全て正方形に分けることができたら、短辺の長さが最大公約数である
   (イ) 余りが１なら、最大公約数は１となる。２つの値は互いに素である（※２つの整数の間に、公約数が１しかない)
   (ウ) あまりが０、１でなければ、余った部分の長方形に注目して、手順の１に戻る

例を用いて説明します。

１つ目の例は、６と３６の最大公約数を求める例です。６と３６の２つの値を、長方形の辺の長さとみなします。図にすると下の通りです。

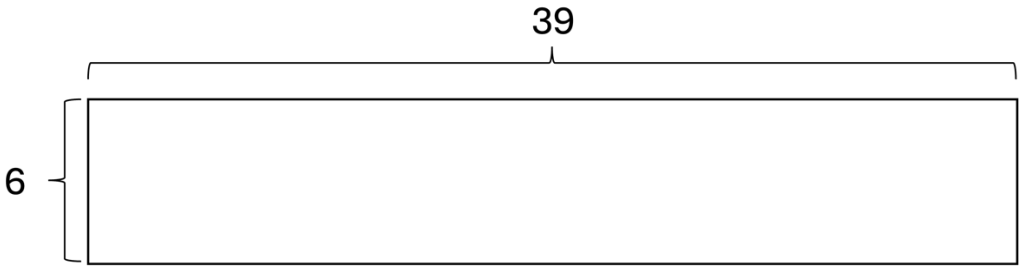


短辺（６）の長さを辺とする正方形で、この長方形を区切っていく（＝長辺（３６）を短辺（６）で割る）と、余りがありません。

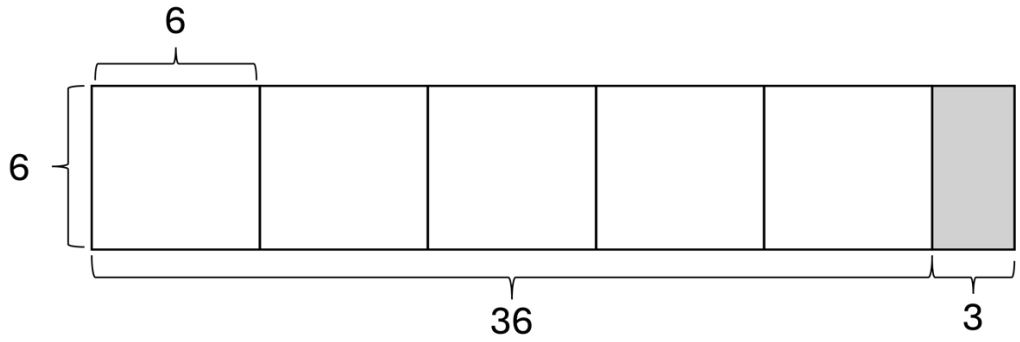


このとき、短辺の長さ（６）が、６と３６の最大公約数になります。

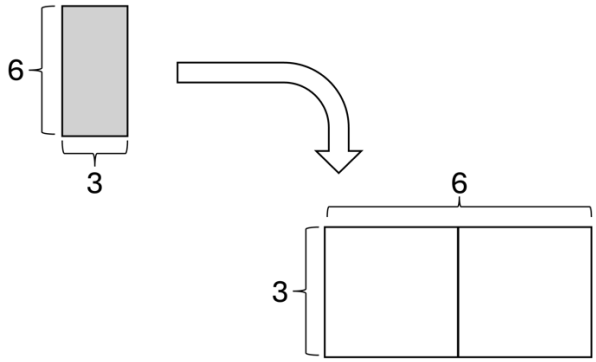
次に、６と３９の最大公約数を求める例を考えます。２つの値を、長方形の辺の長さとみなします。



短辺（６）で長辺（３９）を割ると、余りが生じます。



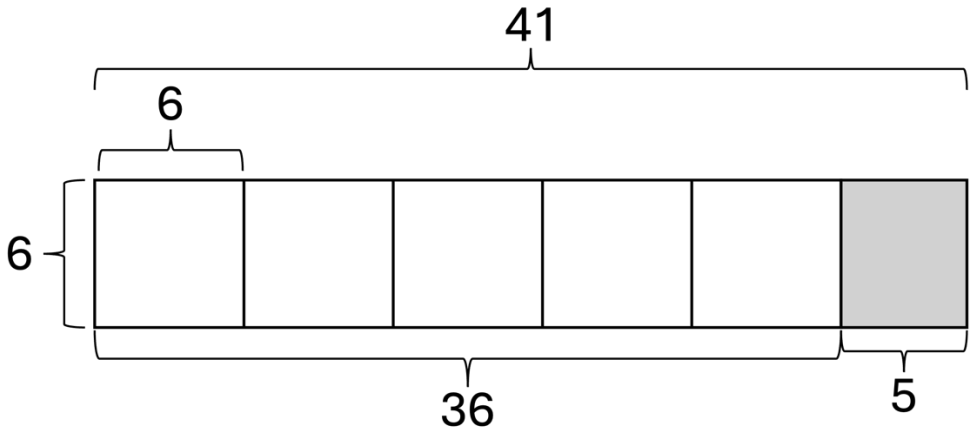
余りの部分に注目すると、長方形と見なせます。



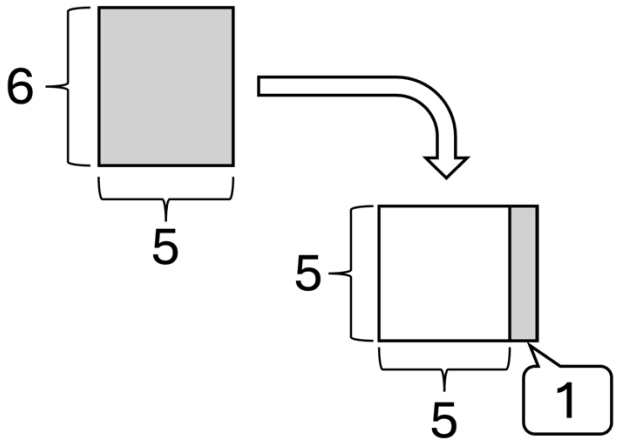
余りの部分の長方形について、短辺（３）の値で長辺（６）を割ります。今度は余りが０になります。
従って、６と３９の最大公約数は３です。

本当に３でいいの？と思うかもしれませんが、１つ前の図で見た通り、他の部分（※１つ前の図で、「36」の部分）は、６で（※１つ前の手順では短辺の長さでした）割り切れます。その「６」を割り切れる「３」なら、他の部分（「36」の部分）も割り切れます。

最後の例は、６と４１です。２つの値を、長方形の辺の長さとみなします。



短辺（６）で長辺（４１）を割った余りは５で、余った部分は短辺５、長辺６の長方形になります。



短辺（５）で長辺（６）を割ると、余りは１です。
この場合、６と４１の間の最大公約数は１ということになり、互いに素であることになります。

以上が、問題「２つの整数の最大公約数を求める」の、１つの解法の手順『ユークリッドの互除法』です。

短辺で長辺を割って、余りを求め、その余りを使って短辺を割って…と、繰り返します。その繰り返しの中で、

- もし、余りが０になったら最大公約数が分かったことになります。
- もし、余りが１なら、最大公約数は１で、２つの整数は互いに素であると分かったことになります。

ユークリッドの互除法は、繰り返しと、条件分岐の組み合わせで出来ています。

以下の問題を解く際は、『JavaScript で学ぶプログラミング入門』第６章 繰り返し（反復）と選択（分岐）の組み合わせ がヒントになります。

問題 関数の定義と利用

プログラムの冒頭で、ユーザーから２つの整数の入力を受け付けます。

```
let a = parseInt( prompt(" 1 つ目の整数を入力してください") );
let b = parseInt( prompt(" 2 つ目の整数を入力してください") );
```

ここで入力された２つの整数について、最大公約数を求めるプログラムを作ります。

1.　入力された値を比較して、大きい方を前の変数に代入する

変数 a と、変数 b には、ユーザーが入力した整数が入っています。
この変数 a,b の値を比較して、値が大きい方が変数 a に、小さい方が変数 b に入っているようにしてください。
※最初から a の方に大きい値が入っていたら、交換する必要はありません。

```
if ( a < b ) {
      temp = a;



}

document.write("大きい値:a = " + a);
document.write("小さい値:b = " + b);
```

2.　大きい値を小さい値で割った、余りを求める

変数 a を変数 b で割った余りを求め、それを変数 amari に代入してください。

```
let amari =
```

3. ユークリッドの互除法を使い、最大公約数を求める

ユークリッドの互除法を使い、最大公約数を求めるプログラムを作ってください。

- 繰り返しを使います。
- 余りが 0 になったとき、割る数が最大公約数です。
  - それ以上計算する必要は無いので、break で繰り返しを終了します。
- 余りが 1 になったとき、２つの数は互いに素(1 しか公約数が無い関係)です。
  - それ以上計算する必要は無いので、break で繰り返しを終了します。

```
while (true) {
      amari =   ;
      if ( amari ==   ) {
            document.write("最大公約数は", b, "です");
            break;
      } else if ( amari ==   ) {
            document.write("最大公約数は 1 で、 2 つの値は互いに素です");
            break;
      } else {
            a =   ;
            b =   ;
}
```

（補足）なぜユークリッドの互除法のような手順を使うのでしょう？

２つの整数の最大公約数は、次の手順でも求めることができます。

1. それぞれの整数の約数を全て挙げる
2. それぞれの約数の中から、共通するものを選ぶ
3. 共通する約数の中から、最大のものを選ぶ

2 桁か 3 桁の、小さい値の最大公約数を求める際には、この手順は簡単で便利です。約数の数はそれぞれ数個か十数個程度だからです。

しかし、「２つの 12 桁の数字」のような、大きな数字同士の最大公約数を求める場合、それぞれの約数の数を数え上げるのも大変です。

ユークリッドの互除法の手順は、大きな値でも、比較的少ない回数の計算で最大公約数を求めることができます。

ある問題を解く手順（アルゴリズム）が、複数あることがあります。手順それぞれに特徴・長所・短所があり、状況に応じて使い分けることで、問題をうまく解くことができます。