

課題：菱形を出力する Javascript プログラムを作成せよ。

ソースコード - diamond.js

```
1. //入力された値に基づいて菱形を描くプログラム diamond.js
2. //入力を促す
3. document.write("菱形図形を書く<br>");
4. num = prompt("出力する菱形の最大長を指定して下さい");
5.
6. //入力が正で奇数なら次に進む、while(!(num % 2) || !(num > 0))でも可。
7. while(!(num % 2 && num > 0))
8.     num = prompt("正の奇数でお願いします m(__)m");
9.
10. //型変換&出力
11. num = Math.floor(num);
12. document.write("最大長が", num, "の菱形 ↓ <BR><BR>");
13.
14. //各種変数の設定
15. sp = "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;";
16. at = "*";
17. center = Math.floor(num / 2) + 1;
18.
19. //ここからが作図アルゴリズム
20. for(i = 1; i <= num; i++) //段数は入力通り
21. {
22.     //Math.abs を使い、菱形の上下対称の性質を表現
23.     for(j = 0; j < Math.abs(center - i); j++)
24.         document.write(sp);
25.     //その段のスペースの数からアスタリスクの数を算出
26.     for(l = 0; l < ((Math.abs(center - j) * 2) - 1); l++)
27.         document.write(at);
28.     //改行、これは以外と忘れる。
29.     document.write("<br>");
30. }
```

プログラム解説：

このプログラムは、入力された正の奇数を最大長とした菱形を HTML ソースに出力するプログラムです。

ソースは HTML に直接記述せず、diamond.js で保存し、HTML ソースから読み込む方式をとりました。

大まかな処理の流れはコメントに記述しているので次ページからアルゴリズムの核心を説明していきたいと思います。

4行目でまずは `prompt()`; で変数 `num` に直接値を代入させる為にプロンプトを出現させています。

7行目でその値が基数かつ正の数であるまで入力を促し続けます。
ここでの条件は否定同士の論理和より論理積の否定を用いた方が美しい感じがしたので

```
7. while(!(Math.floor(num) % 2 && num > 0))
```

としました。
正の奇数ということですが、ここでは `Math.floor` を用いて、整数型に変換して正の奇数ならば良いことにしました。

15~17行目で出力する文字を変数に代入、ここで、HTML ソースではスペース文字は無視されるのでスペース文字の代わりに半角スペースを表す ` ` を使用しました。

変数 `center` は菱形の中心のアスタリスクの座標を表しています。
ここで注目して欲しいのが `0.5` を加えていること。
Javascript では変数の扱いが非常に用意で、型宣言も必要ありません。
しかし、型宣言が不要という特徴の悪い面が出てきてしまいました。
ここでは整数 `2` で整数の `num` を除算しているため、`center` も整数型になると C 言語に慣れていた自分は思っていました。しかし、実際には `center` は `num / 2` の結果が `float` 型なら `float` 型となっていました。
そこで変数 `num` の時と同じく `Math.floor()`; で、変数を整数型に型変換しました。

型変換が不要というのは最初は便利！と思いましたが、C 言語に慣れているのであった方がいいかもと思いました。やっぱり C はいいな。。。

次ページから、このプログラムの要、菱形図形作図アルゴリズムを説明していきます。

```

      *              *              *              *              *
     ***            ***            ***            ***            ***
    *****        *****        *****        *****        *****
   *****      *****      *****      *****      *****
  *****    *****    *****    *****    *****
 *****  *****  *****  *****  *****
***** ***** ***** ***** *****
 ***** ***** ***** ***** *****
  ***** ***** ***** ***** *****
   ***** ***** ***** ***** *****
    ***** ***** ***** ***** *****
     ***** ***** ***** ***** *****
      *              *              *              *              *
```

*****作図アルゴリズム (プログラムから抜粋)*****

```
1. for(i = 1;i <= num;i++)
2. {
3.   for(j = 0;j < Math.abs(center - i);j++)
4.     document.write(sp);
5.   for(l = 0;l < ((Math.abs(center - j) * 2) - 1);l++)
6.     document.write(at);
7.   document.write("<br>");
8. }
```

アルゴリズム説明：

まず、1行目のfor(i = 1;i <= num;i++)は、C言語でもおなじみのfor文による繰り返しである。

ここで、for(i = 0;i < num;i++)記述することも出来たが(この方がオーソドックスかも)ここで、iは現在文字を出力中の段落を表す変数として使用するため、1から初めている。

そして3、4行目の

```
for(j = 0;j < Math.abs(center - i);j++)
  document.write(sp);
```

は、段落にスペースを出力するモジュールである。

ここではコメントで記述しているとおり、菱形は上下対称(左右対称でもあるがここではその性質は利用していない。)なので、centerを基準に現在の段落がどれだけcenterから離れているかを基準に、スペースの数を決めていいる。(ここでは半角スペース2つ分を1つの変数として扱っている。)

真ん中の段落はスペースは必要なく、真ん中の段落から1段離れるごとにスペースが1つずつ増えていく性質をMath.abs();を利用して上手く表現出来たと思う。

さらに5、6行目の

```
for(l = 0;l < ((Math.abs(center - j) * 2) - 1);l++)
  document.write(at);
```

は、段落に菱形を形作るアスタリスクを出力するモジュールである。

Math.absを利用している意図はスペース出力モジュールと同様である。

ここでスマートな点は、段落から出力するアスタリスクの数を算出するのではなく、直前で解っているスペースを出力した回数から算出しているところである。

簡単に言えばスペースが多いほどアスタリスクは少なく、スペースが少ないほどアスタリスクは多い。

その性質と、 $2K - 1$ という奇数の式を利用綺麗に利用したアルゴリズムになったと思う。

そして最後に、

```
document.write("<br>");
```

で段落を次の行に進めている。ここは以外と忘れる部分である。

