

ゼロの効用 → 8x9Craft はおもしろい

2018.3.25 CoderDojo Nada 辻

なにもないことを表すゼロ

ゼロを使って表す: 103

ゼロを使わないで: 1 3 ← わかりづらい、13 と間違える

10進法

- ・ 使う数字は、0,1,2,3,4,5,6,7,8,9 の 10 種類

3502

- ・ 3 は 1000 が 3 個ある
- ・ 5 は 100 が 5 個ある
- ・ 0 は何もないことを表す
- ・ 2 は 1 が 2 個ある

$$3 \times 1000 + 5 \times 100 + 0 \times 10 + 2 \times 1$$

$$3 \times 10^3 + 5 \times 10^2 + 0 \times 10^1 + 2 \times 10^0$$

2進法

使う数字は、0,1 の2種類

1101

- 先頭の1は1000（10進法で8）が1個ある
- 2番目の1は100（10進法で4）が1個ある
- 3番目の0は何もないことを表す
- 4番目の1（10進法で1）が1個ある

$$1 \times 1000(8) + 1 \times 100(4) + 0 \times 10(2) + 1 \times 1(1)$$

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

漢数字で3502を表す






三千五百二（さんぜんごひゃくに）

三千五百〇二（さんぜんごひゃくくうに）

○や□などの文字をはさんで表す場合もあった

○や□などの文字は「空（くう）」と呼ばれた






10 のゼロ乗とは何か？

$10^2 =$	100	10 を 2 回かける		10 分の 1
$10^1 =$	10	10 を 1 回かける		10 分の 1
$10^0 =$	1	10 を 0 回かける？		10 分の 1
$10^{-1} =$	$\frac{1}{10}$	10 を -1 回かける？		10 分の 1
$10^{-2} =$	$\frac{1}{100}$	10 を -2 回かける？		10 分の 1

※ 10^0 があることで、きれいに並ぶ

※ 10^0 があることで、パターン化される、ルールを 1 本にできる

2 のゼロ乗とは何か？

$2^2 =$	4	2 を 2 回かける		2 分の 1
$2^1 =$	2	2 を 1 回かける		2 分の 1
$2^0 =$	1	2 を 0 回かける？		2 分の 1
$2^{-1} =$	$\frac{1}{2}$	2 を -1 回かける？		2 分の 1
$2^{-2} =$	$\frac{1}{2}$	2 を -2 回かける？		2 分の 1

※ 2^0 があることで、きれいに並ぶ

※ 2^0 があることで、パターン化される、ルールを 1 本にできる

ゼロの役割と効果

- 「ない」けど場所はとっておく

間隔を空ける: 1 1

ゼロを埋める: 1001 ← こっちのほうがわかりやすい

- 式を一般化できる $\sum a_k \times 10^k$

(例) 3502 $3 \times 10^3 + 5 \times 10^2 + 0 \times 10^1 + 2 \times 10^0$ $\sum_{k=0}^3 a_k \times 10^k$ $a=\{3,5,0,2\}$

- 形をそろえて、扱いやすくする

(例) いろんな形の荷物を決まった形のコンテナに入れて、処理を単純化



四角の箱、ブロックといえは？

MineCraft のブロックでゼロとは？

ゼロは空気です。



石を十字の形に置く

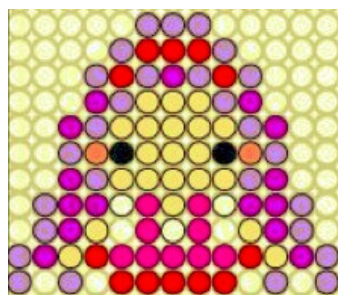


簡単に置きたい

十字のタテ・ヨコのラインをそれぞれ繰り返すか？

位置決め、繰り返しの回数の設定がめんどろ！

パーラービーズのやり方でやってみる。



空気と石の配列（リスト）を作る

```
let drw = [  
  [0, 0, 1, 0, 0],  
  [0, 0, 1, 0, 0],  
  [1, 1, 1, 1, 1],  
  [0, 0, 1, 0, 0],  
  [0, 0, 1, 0, 0],  
];
```

空気を入れることで、5×5の配列になる

2次元配列を回して、ブロックを並べる

```
for (let y = 0; y < 5; y++) {  
  for (let x = 0; x < 5; x++) {  
    let d = drw[y][x]  
    // ブロック番号、メタ番号、X座標、Y座標、Z座標  
    world.setBlock(d, 0, x, 5 - y, 1)  
  }  
}
```

効果！

同じ大きさの絵であれば、リスト（配列）の中身だけ変えれば、違う絵が作れる

```
let drw = [  
  [1, 0, 0, 0, 1],  
  [0, 1, 0, 1, 0],  
  [0, 0, 1, 0, 0],  
  [0, 1, 0, 1, 0],  
  [1, 0, 0, 0, 1],  
];
```



ゼロ（空気）をセットすることで、処理が簡単になった！

JavaScript の関数で空 (ゼロ) を使う

空 (何もしない関数) を使ったら便利になるか？

まずは、空を使わずに

```
<html>
<head>
  <meta charset="UTF-8">
  <script>
    function init() {
      // 配列初期化
      let a = [];
      for (let i = 0; i < 5; i++) {
        a[i] = {};
      }

      // 配列の個々の 0,2,4 番目の要素だけに test 関数を定義
      a[0].test = function() {document.writeln("<p>1 目関数から書き出し </p>");};
      a[2].test = function() {document.writeln("<p>3 目関数から書き出し </p>");};
      a[4].test = function() {document.writeln("<p>5 目関数から書き出し </p>");};

      // 配列の 0,2,4 番目の要素に対して、test メソッドを実行
      for (let i = 0; i < 5; i++) {
        if (i % 2 === 0) {
          a[i].test();
        }
      }
    }
  </script>
</head>
<body onload="init()">
</body>
</html>
```

処理結果

1 目関数から書き出し

3 目関数から書き出し

5 目関数から書き出し

空（何もしない関数）を使ってみる

```
<html>
<head>
  <meta charset="UTF-8">
  <script>
    function init() {
      // 配列初期化
      let a = [];
      for (let i = 0; i < 5; i++) {
        a[i] = {};
      }

      // 配列の個々の要素に test 関数を定義
      a[0].test = function() {document.writeln("<p>1 回目の関数から書き出し </p>");};
      a[1].test = function() {}; // 空の関数をセット
      a[2].test = function() {document.writeln("<p>3 回目の関数から書き出し </p>");};
      a[3].test = function() {}; // 空の関数をセット
      a[4].test = function() {document.writeln("<p>5 回目の関数から書き出し </p>");};

      // 配列の全部の要素に対して、test メソッドを実行
      a.forEach(function(e) {
        e.test()
      });
      // もっと簡潔に（関数を式で書く）
      a.forEach(e => {e.test();});
    }
  </script>
</head>
<body onload="init()">
</body>
</html>
```

要素を区別しなくてよくなった。

処理が単純になった！

for 文で回す回数や、条件文の記述ミスが起きない！

では、最後におまけ

このリスト（配列）を描くと何になるでしょうか？

では、MineCraft(8x9Craft)で！

[illegible]

おしまい

