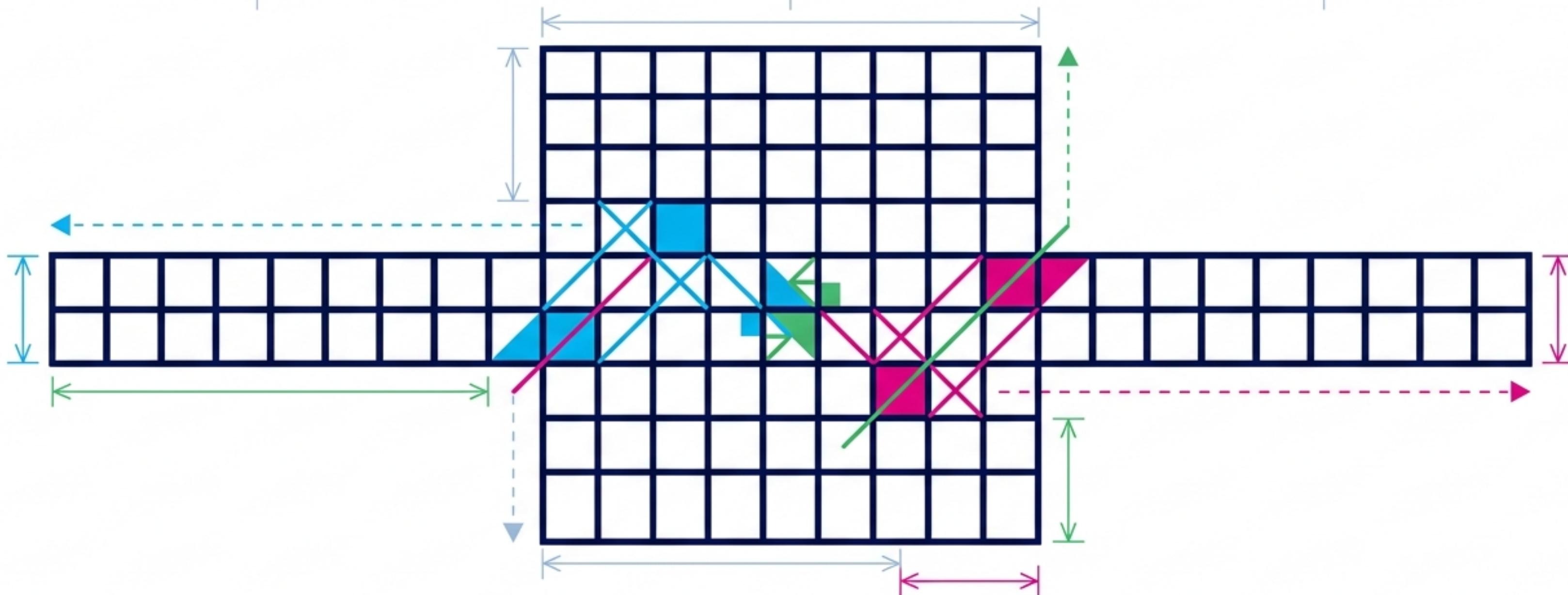




設問2を視覚的に解き明かす：重複チェックのアルゴリズム

令和4年度 秋期 応用情報技術者試験 午後 問3 プログラミング



※本スライドでは「自力で正解を導くための思考プロセス」を解説します。解答そのものは記載していません。パズルを解くように、論理のピースを組み立ててください。

```

function row_ok(n, x)                                //横 1 行の重複チェック
    row_top ← オ              //行の左端のマスを示す添字を求める
    for (i を 0 から 8 まで 1 ずつ増やす)
        if ( カ )
            return false
        endif
    endfor
    return true
endfunction

function column_ok(n, x)                             //縦 1 列の重複チェック
    column_top ← キ           //列の上端のマスを示す添字を求める
    for (i を 0 から 8 まで 1 ずつ増やす)
        if ( ク )
            return false
        endif
    endfor
    return true
endfunction

function frame_ok(n, x)                             //3x3の枠内の重複チェック
    frame_top ← x - ケ - mod(x, 3) //枠内の左上端のマスを示す添字を求める
    for (i を 0 から 2 まで 1 ずつ増やす)
        for (j を 0 から 2 まで 1 ずつ増やす)
            if (board[frame_top + 9 * i + j]がnと等しい)
                return false
            endif
        endfor
    endfor
    return true
endfunction

```

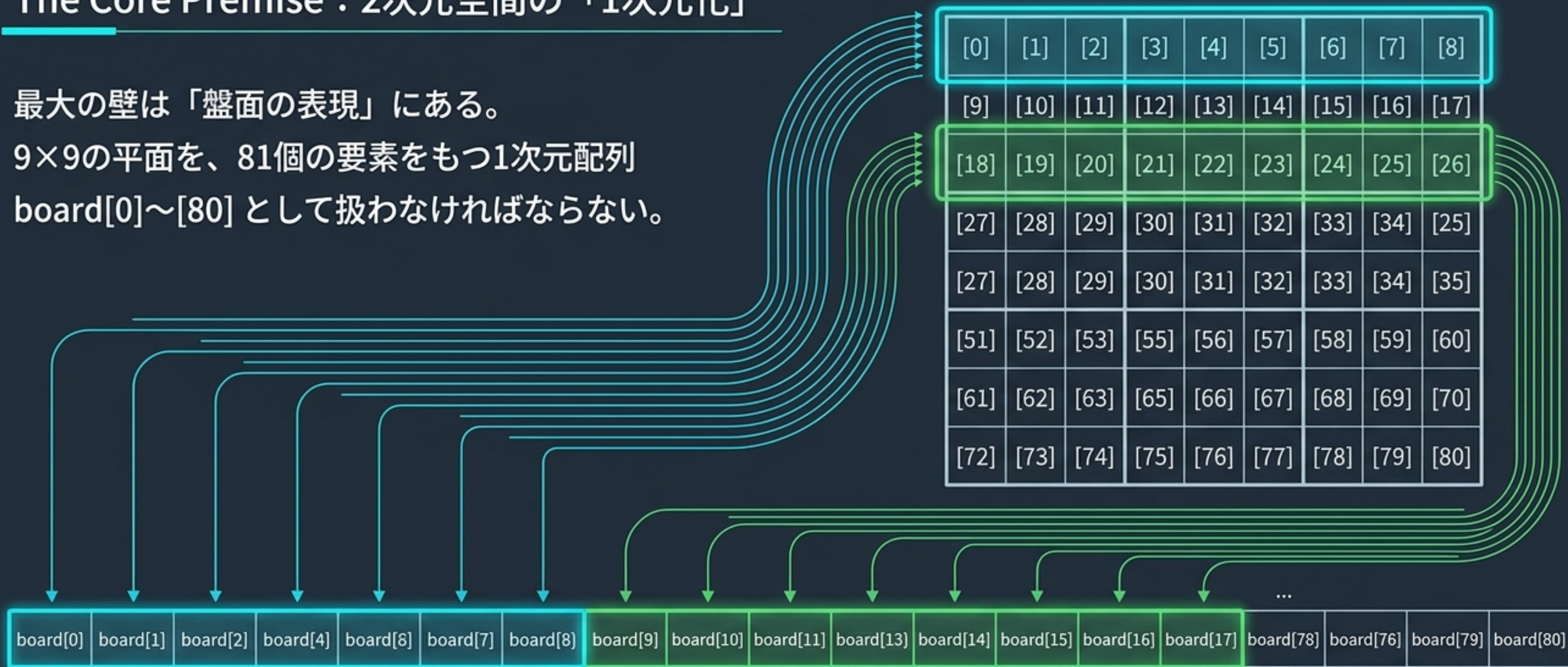
The Goal：パズルのルールを数式に翻訳する

2	8	1	4	9	3	7	6	5
9	4	6	2	5	7	3	8	1
5	7	3	1	6	8	4	2	9
4	9	5	6	7	1	2	3	8
6	1	8	3	2	5	9	7	4
3	2	7	8	4	9	5	1	6
7	6	2	9	8	4	1	5	3
1	5	9	7	3	6	8	4	2
8	3	4	5	1	2	6	9	7

- ミッション: 重複チェック関数 (row_ok, column_ok, frame_ok) の空欄を完成させる。
- 制約: 縦・横・3x3枠内に同じ数字が存在してはならない。
- 課題: コンピュータに「空間の広がり」をどう計算させるか？

The Core Premise : 2次元空間の「1次元化」

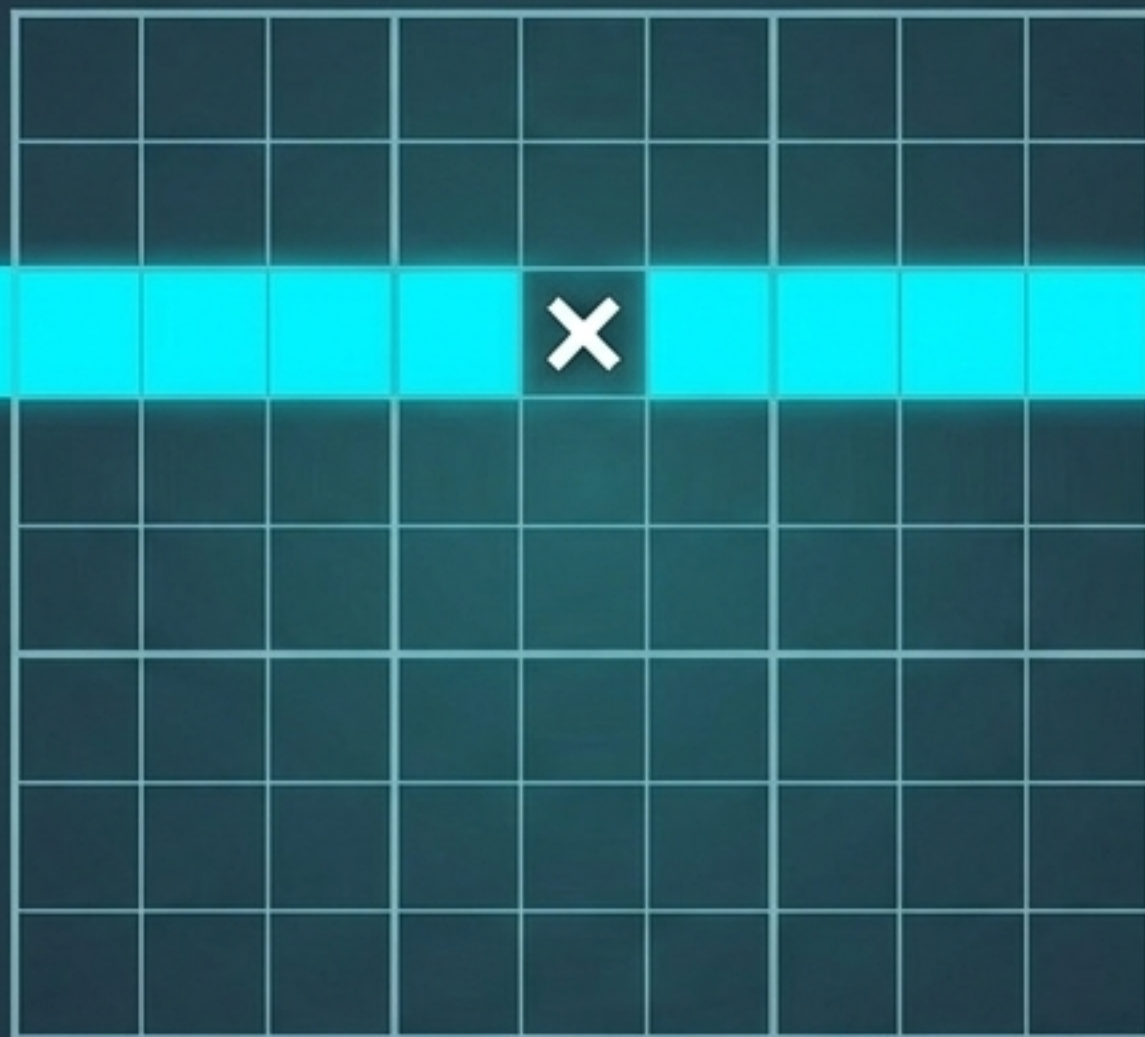
最大の壁は「盤面の表現」にある。
9×9の平面を、81個の要素をもつ1次元配列
board[0]~[80] として扱わなければならない。



問い：「対象のマス x」から、どうやって「行の先頭」「列の先頭」「枠の先頭」を数式で割り出すか？

Chapter 1: The Row Check (row_ok)

マス x と同じ「横1行」に、数字 n が既に存在しないかチェックするアルゴリズムを組み立てる。

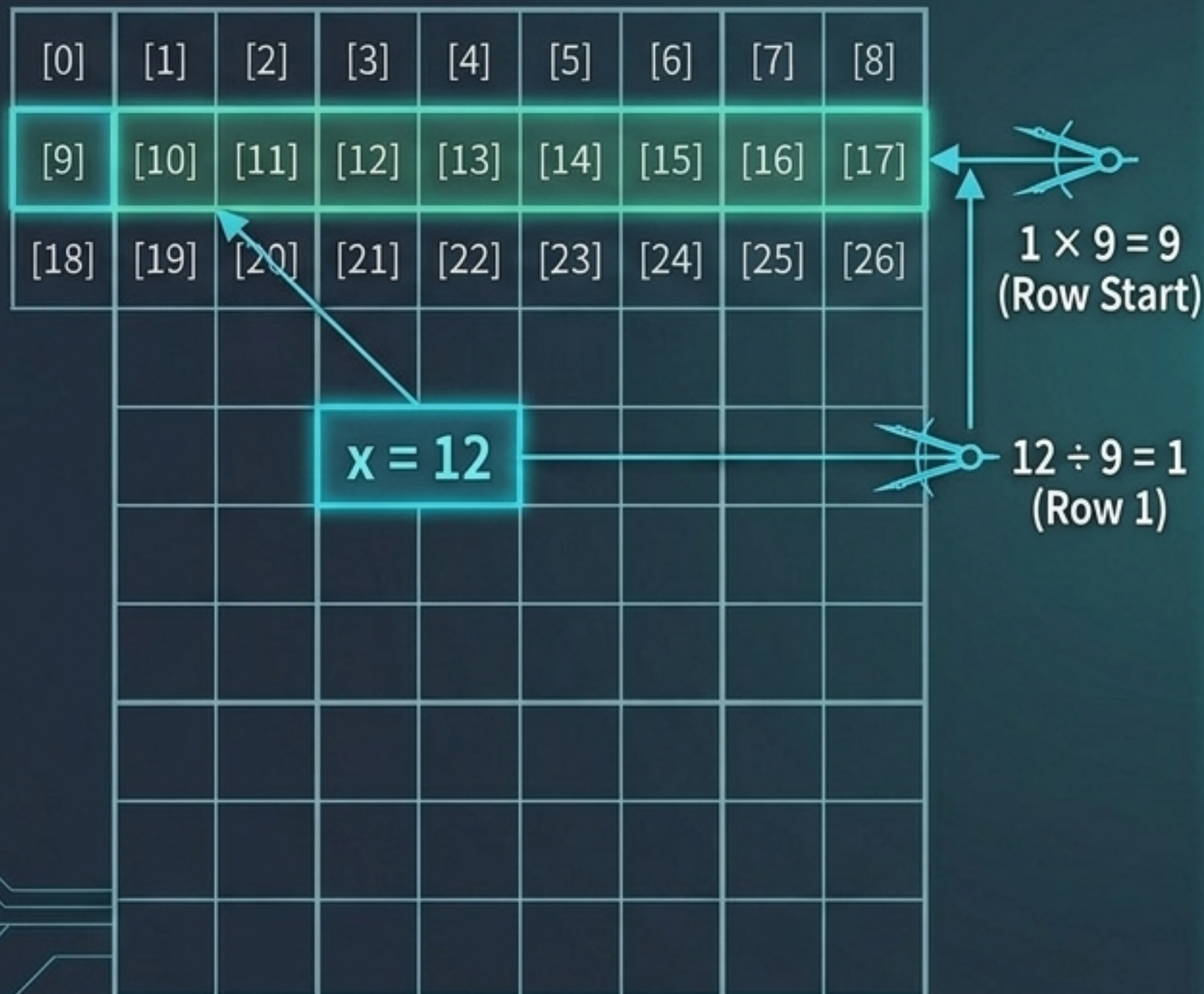


Step 1: 左端のマス (row_top) を特定する
→ 【空欄 才】

Step 2: 左から右へ9マス分をスキャンする
→ 【空欄 力】

Deducing [オ] : 行の先頭を見つける (Row Snapping)

幾何学的空間計算 (Visual Math)

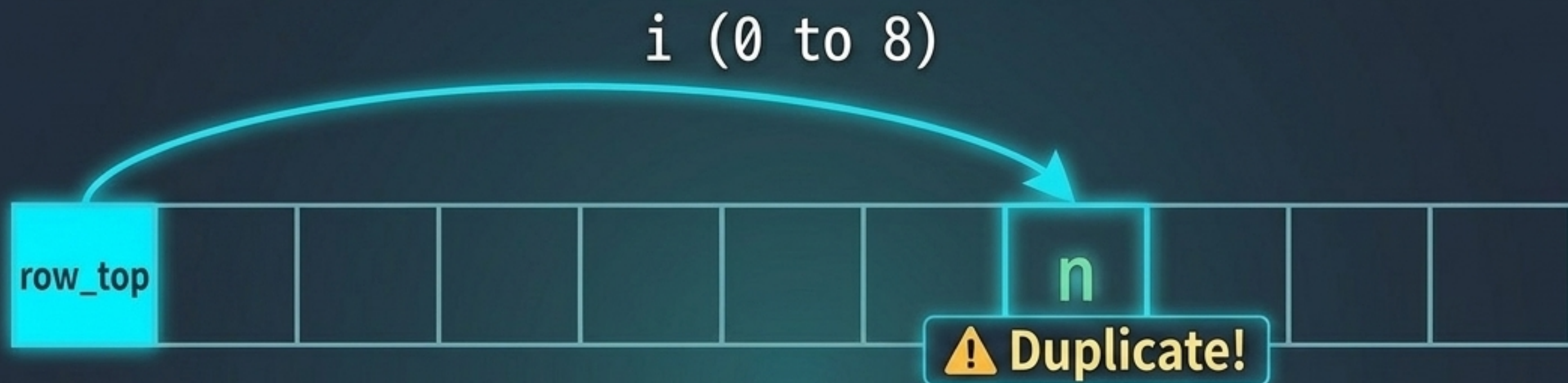


コード翻訳 (Code Translation)

思考プロセス: x がどの行に属しているか？
盤面は横に「9マス」ずつ区切られている。
 x を「1行のマス数 (9)」で割った商 (div関数) を求めることで、行の番号 (Y座標) が分かる。
その行の「左端の添字」に戻すには、求めた商に何を掛ければよいだろうか？

```
row_top = [ オ ]  
div(x, 9) * [ ? ]
```

Deducing [力] : 行のスキャンと条件判定



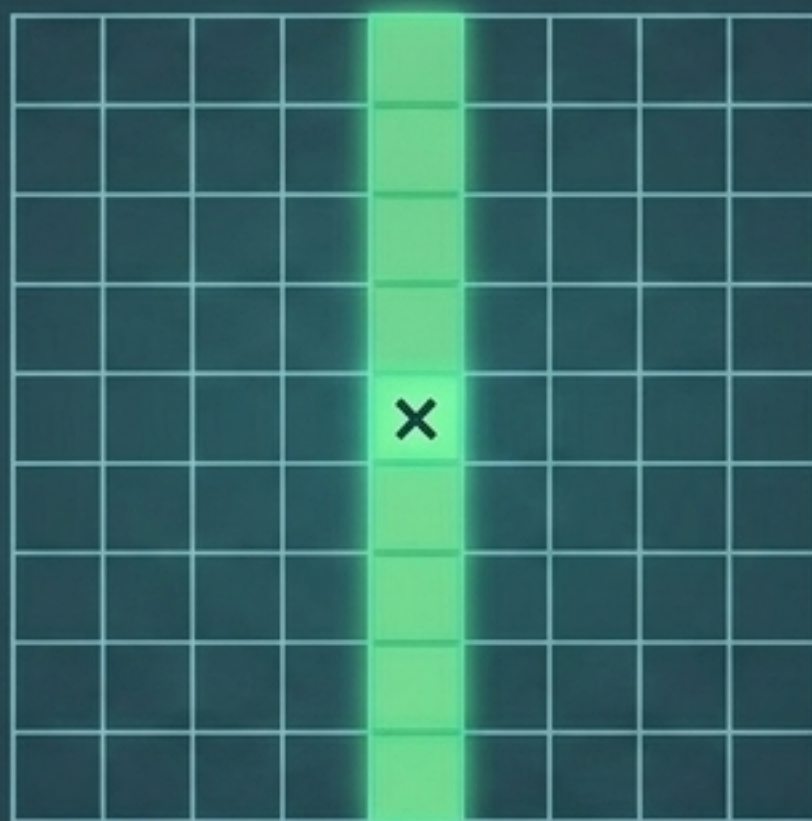
```
for (i を 0 から 8 まで 1 ずつ増やす) {  
    if ( [ 力 ] ) return false  
}
```

思考プロセス: 左端 (row_top) から右へどう移動するか?

- ループ変数 i が 0 から 8 まで増加する。
- 調べるべきマスは、左端の添字に i を足した位置。
- 問: そのマスの値が、入れようとしている数字 n と「等しい」場合を弾くには、どう条件式を記述すべきか?

Chapter 2: The Column Check (column_ok)

マス x と同じ「縦1列」に、数字 n が既に存在しないか
チェックするアルゴリズムを組み立てる。



Step 1: 上端のマス (column_top) を特定する → 【空欄 キ】



Step 2: 上から下へ9マス分をスキャンする → 【空欄 ク】

Deducing [≠] : 列の上端を見つける (Column Snapping)

Visual Math

7

x = 34

$$34 \div 9 = 3 \\ \text{余り } 7$$

Code Logic

思考プロセス: x が左から何列目にあるか?

- 各列の一番上のマスの添字は、必ず 0 ~ 8 のいずれかになる。
- 1次元配列において、全体の横幅「9」で割った余りは、そのまま「X座標 (列の上端)」を表す。

問: 対象のマス x を使って、その上端の添字を割り出す式は?

```
column_top = [ ≠ ]
```

```
mod(x, ?)
```

Deducing [ク] : 列のスキャンと縦移動の法則

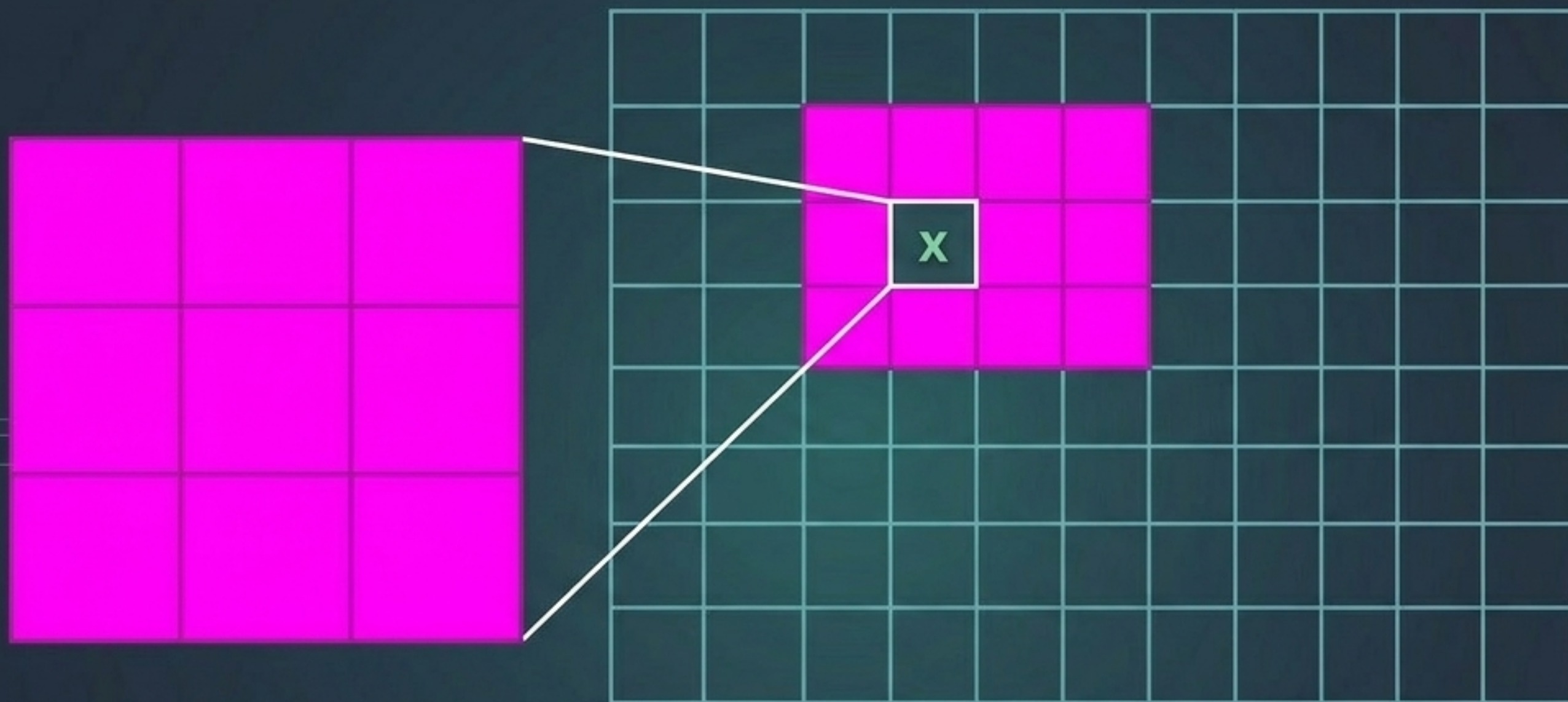


```
for (i を 0 から 8 まで 1 ずつ増やす) {  
    if ( [ク] ) return false  
}
```

思考プロセス: 1次元配列の中で「真下」へ進むには？

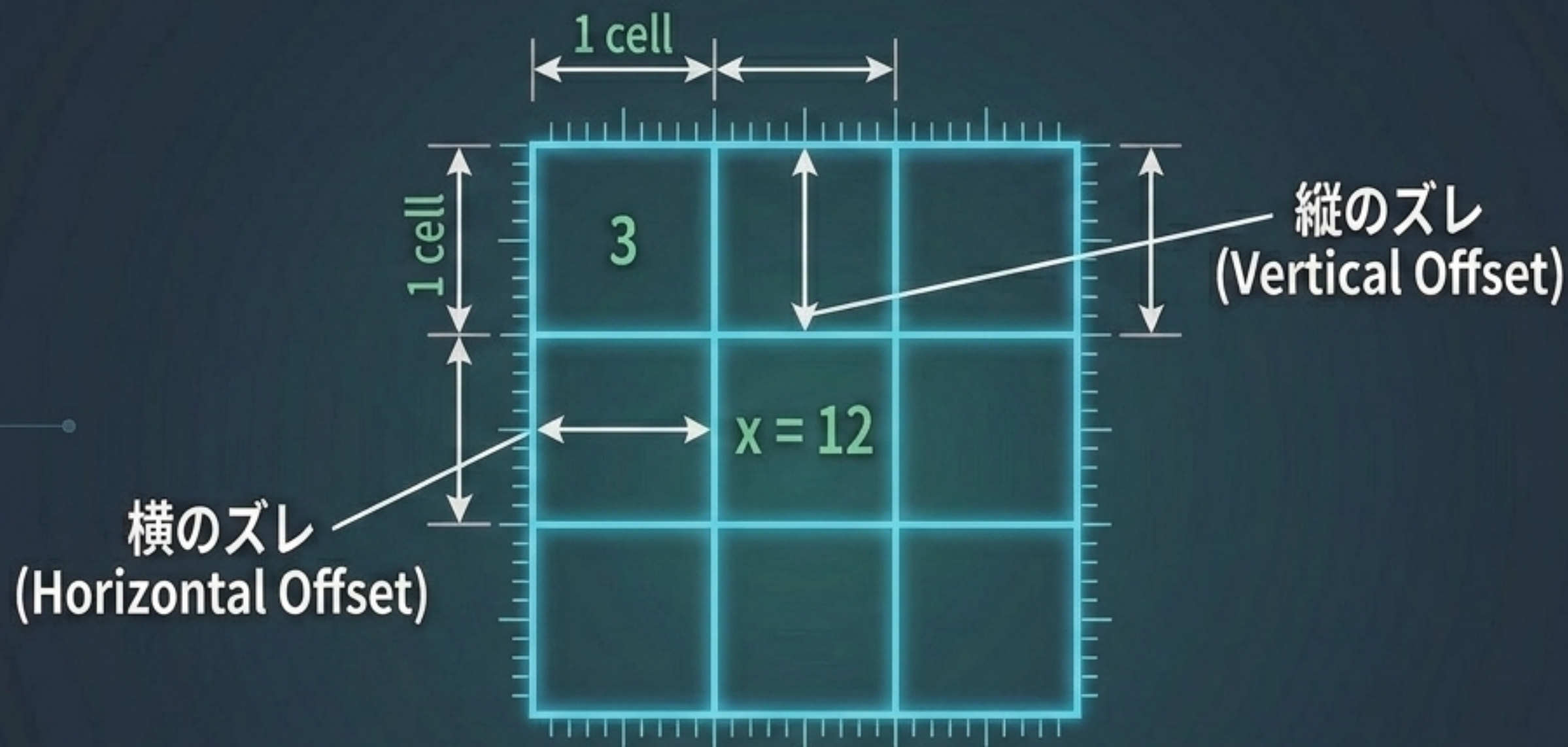
- 1行は9マス。つまり、真下のマスへ移動するにはインデックスを「9」増やす必要がある。
- ループ変数 i を使って、「 i 行分下へ移動した位置」の添字を計算する。
- **問:** 上端のマスに、どのような計算式を足せば縦方向のスキャンになるか？そして、その値が n と等しいかをどう判定するか？

Chapter 3: The Block Check (**frame_ok**)



- **目的**: マス x が含まれる「3×3の枠内」の重複チェック。
- **最大の難関**: 枠内の左上端のマス (frame_top) を特定するためのBの数式のピース【**空欄ケ**】を解読する。

The Anatomy of a 3x3 Block : 2つの「ズレ」を引く



枠の左上端に戻るための唯一の方法。現在のマス x から、 3×3 枠内における「横のズレ」と「縦のズレ」の両方を引き算して、原点（左上角）に巻き戻すという空間操作が必要になる。

Formula Deconstruction : 未知のピースの正体

$$\text{frame_top} = x - [\text{ケ}] - \text{mod}(x, 3)$$

枠内の縦のズレを消去
(Removes Y-offset)

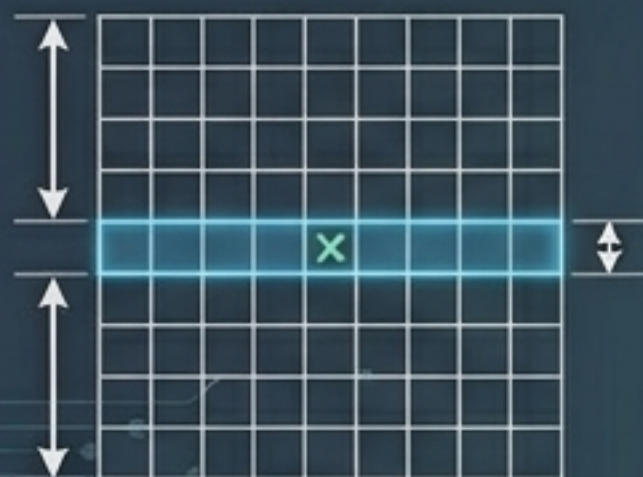
枠内の横のズレを消去
(Removes X-offset)

公式の解剖 : $\text{mod}(x, 3)$ は、3マスの幅の中で自分が左から何番目か（横のズレ）を求めている。これを引くことで、縦のラインが揃う。

結論 : ならば、残された未知のピース【空欄 ケ】は、枠内での「縦方向のズレ」を求めて差し引くための数式であるはずだ。

Deducing [ケ]: 縦のズレを数式化する

1. 絶対的な行を求める



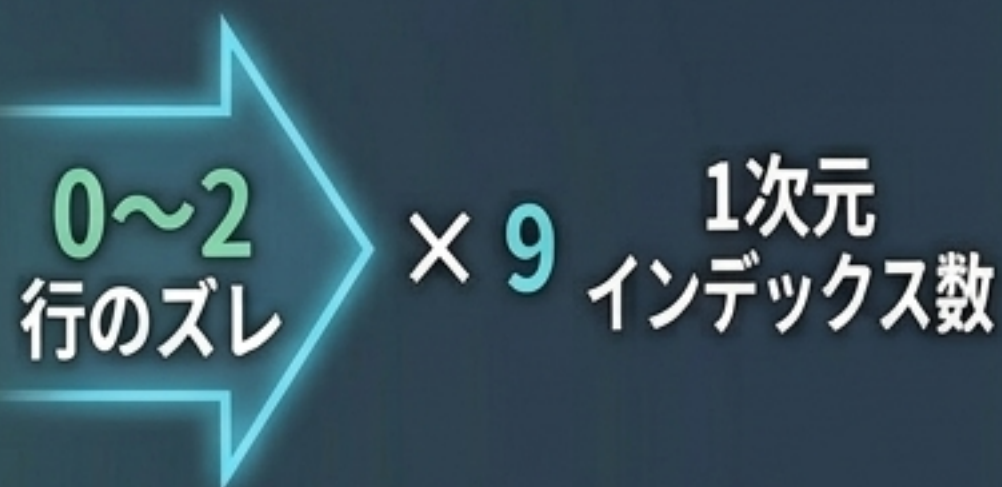
xが盤面全体の何行目か？
row_okで使った
 $\text{div}(x, 9)$ を利用する。

2. 枠内の相対的な行を求める



その行が「3行ごとのブロック」
の中で上から何番目か？
「3」で割った余り
 $\text{mod}(?, 3)$ をとる。

3. 1次元距離に換算する



求めた「0~2行のズレ」を、
実際の1次元インデックス数に
戻すため、幅である「9」
を掛ける。

数式の組み立て: $\text{mod}(\text{div}(x, 9), 3) * [?]$

Synthesis : 1次元配列を2次元空間として歩く3つの原則

横方向の移動 (X-axis)

- 始点：幅(9)で割った「商 (div)」
- スキャン：インデックスに+1していく。

縦方向の移動 (Y-axis)

- 始点：幅(9)で割った「余り (mod)」
- スキャン：インデックスに+幅(9)していく。

枠内の移動 (Micro-Grid)

- 始点：div と mod を組み合わせて縦横の相対ズレを抽出し、現在地から引く。

この3つの数学的原則を組み合わせることで、[オ]~[ケ]のすべての空欄を自力で完全に埋めることができる。

Beyond the Exam : 空間を翻訳する力

0	1	3	6	5	4	7	8	9
7	5	2	3	4	6	8	1	2
5	3	1	7	8	1	2	6	4
4	6	8	1	2	9	5	3	7
3	7	9	6	3	5	1	2	5
1	2	3	5	7	4	7	6	9

```
grid[row][col] = value;  
int boxIndex = (row/3)**3 + col/3;  
if (isValid(board, row, col, num)) {  
    return row, num3;  
}
```

- コードの空欄補充は、単なる暗記ではなく「空間の翻訳作業」です。
- divとmodが、1次元の直線上になぜ「縦・横・枠」の概念を生み出せるのか。その本質を理解すれば、どのような探索アルゴリズムも読み解くことができます。
- それでは、解説した思考プロセスを基に、あなた自身の力で数式を導き出してください。