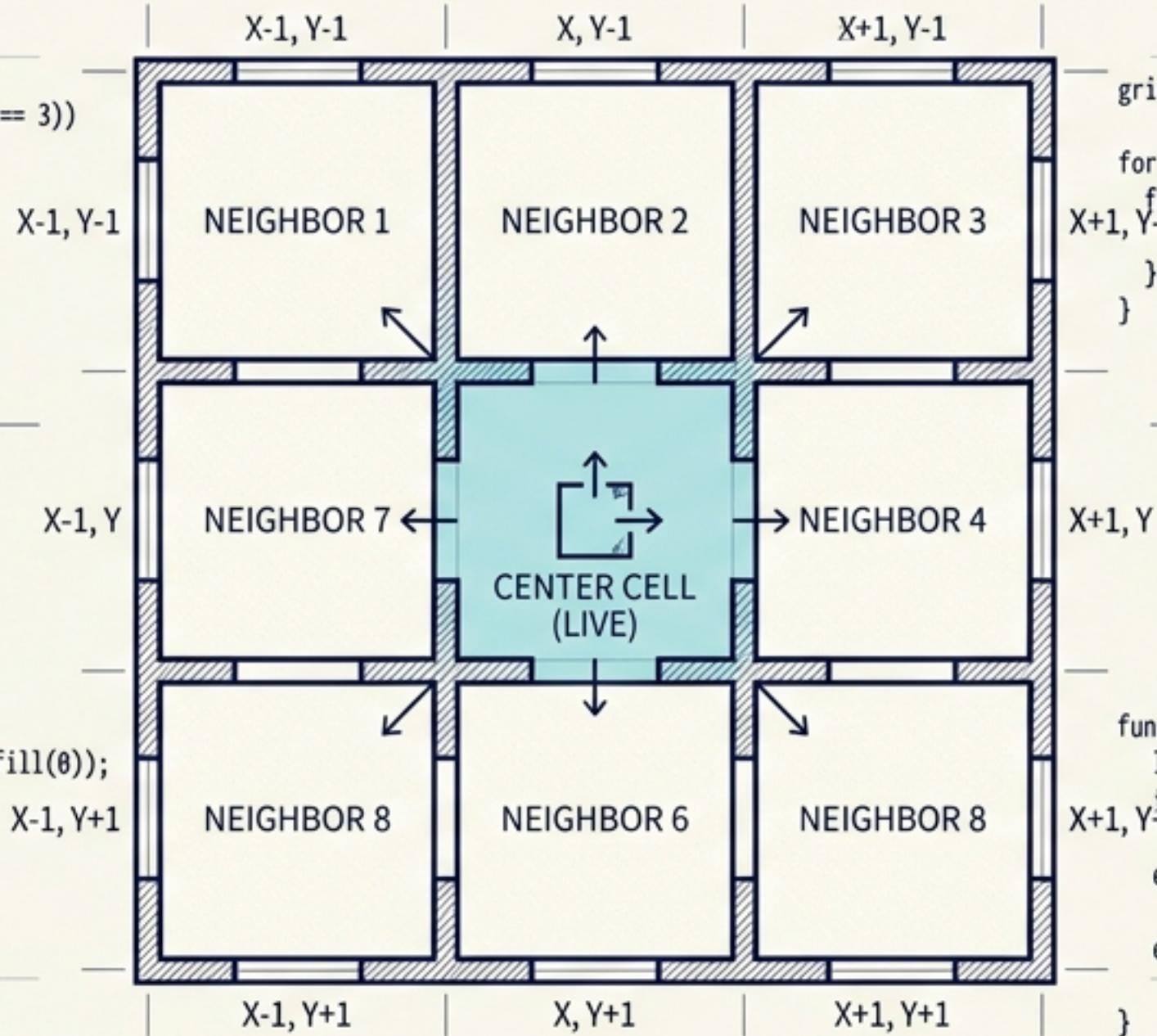


ライフゲーム解読の設計図

応用情報技術者試験「午後問3」を自力で突破するための思考ガイド

```
function updateCell(x, y) {  
  let neighbors = countNeighbors(x, y);  
  if (grid[x][y] === 1 && (neighbors === 2 || neighbors === 3))  
    return 1;  
  } else if (grid[x][y] === 0 && neighbors === 3)  
    return 1;  
  } else  
    return 0;  
}
```

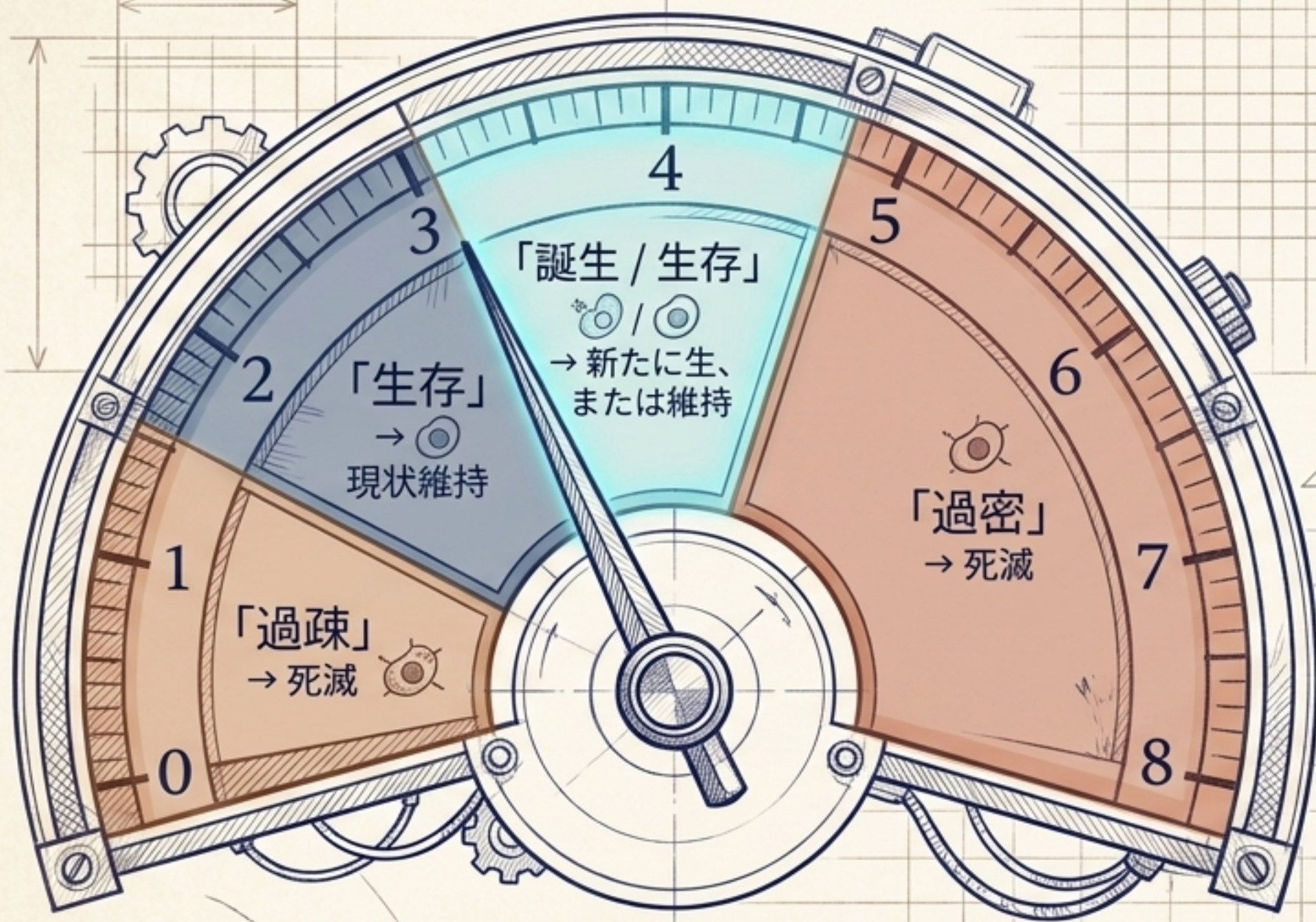
```
grid = new Array(rows).fill(0).map(() => new Array(cols).fill(0));  
for (let i = -1; i <= 1; i++) {  
  for (let j = -1; j <= 1; j++) {  
    if (i === 0 && j === 0) continue;  
  }  
}
```



```
grid = new Array(rows).fill(0).map(() => new Array(cols).fill(0));  
for (let i = -1; i <= 1; i++) {  
  for (let j = -1; j <= 1; j++) {  
    if (i === 0 && j === 0) continue;  
  }  
}
```

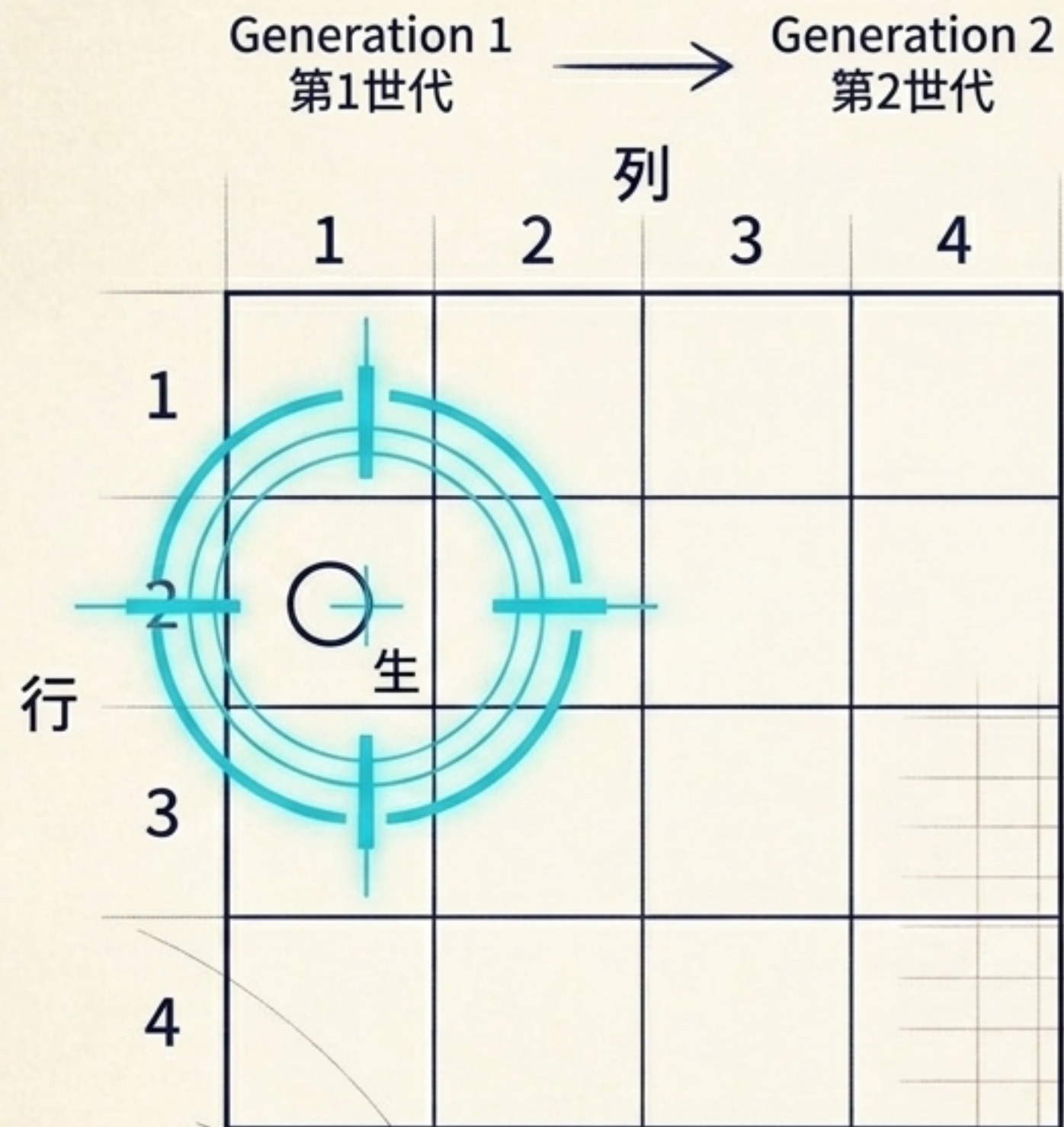
```
function updateCell(x, y) {  
  let neighbors = countNeighbors(x, y);  
  if (grid[x][y] === 1 && (neighbors === 2 || neighbors === 3))  
    return 1;  
  else if (grid[x][y] === 0 && neighbors === 3)  
    return 1;  
  else return 0;  
}
```

※本スライドに解答の直接的な記載はありません。あなた自身で正解を導き出すための「視点」を提供します。



ルール適用の絶対条件

- ✓ チェックポイント①：
現在のマスは「生」か「死」か？
- ✓ チェックポイント②：
周囲の「生」のマスはいくつあるか？



STEP 1

ターゲットマスの現在の状態（生／死）を把握する。

STEP 2

周囲8マス（斜めを含む）をスキャンし、「生（○）」の数をカウントする。
（※ここを見落とさないように！）

STEP 3

カウント数と前スライドのメーターを照らし合わせる。

Q:「[ア]のマスは現在『生』。周囲の『生』はいくつ？
メーターのどの領域に該当するだろうか？」

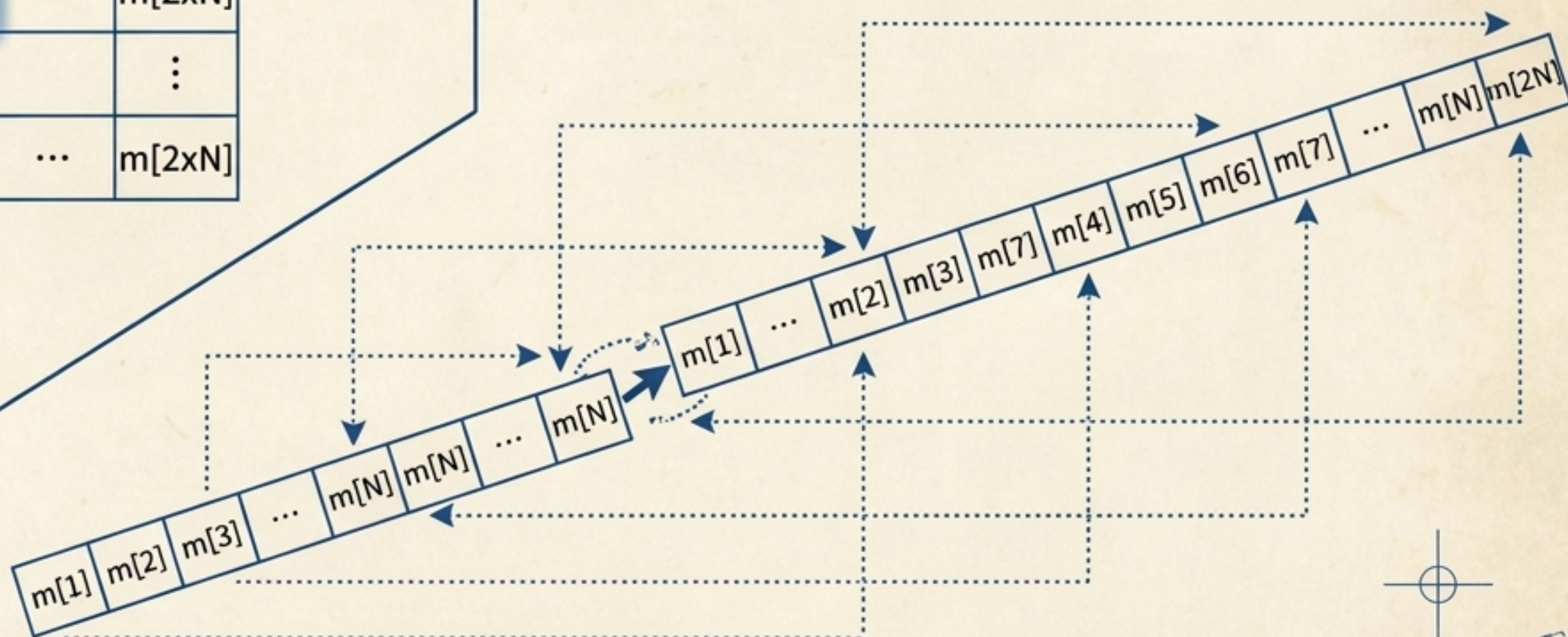
行

列

	1	2	...	j	...	N
1	m[1]	m[2]	...	m[3]	...	m[N]
2	m[1]	m[2]	...	m[5]	...	m[2xN]
⋮	⋮	⋮		⋮		⋮
k	m[k1]	m[k1]	...	m[オ]	...	m[2xN]
⋮	⋮	⋮		⋮		⋮
N	m[N+1]	m[N+2]	...	m[N+1]	...	m[2xN]

オフセット (行のスキップ) : k行目に到達するまでに、完全に通過した行はいくつある? (ヒント : k-1行分)。その中に含まれるマスの総数は?

端数 (列の移動) : 目的の行に到達した後、左から何番目へ進む?



Q: 「通過済みの『ブロックの束 (掛け算)』と、現在の行での『歩数 (足し算)』を組み合わせると、どんな式になる?」

copy(m, temp)

現在の世代のバックアップ

temp

1	0	1	1	1	0	0	0	1	0
0	1	1	0	0	1	0	1	0	1
1	1	1	1	1	1	0	1	0	0
1	0	0	1	0	1	1	1	0	1
1	1	1	0	1	1	0	0	0	1
1	1	0	0	1	0	1	1	0	1

clear(m)

次世代の書き込み先

m

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Q: 「いま周囲の『生のマス』を数えようとしている。カウント対象（判定基準）として参照すべき配列は、初期化されて真っ白な m だろうか？それとも退避させた temp だろうか？」

ビジネスロジック (Rules)

Panel 1

ルールA：現在のマスが「死」かつ、
周囲が「3」なら「誕生 (1)」

Panel 2

ルールB：現在のマスが「生」かつ、
周囲が「2 or 3」なら「生存 (1)」

コードブロック (Code)

Snippet 1

```
if ( [ キ ] ) {  
    m[i] ← 1  
}
```

Snippet 2

```
elseif ( temp[i]==1 and  
        (e==2 or e==3) ) {  
    [ ク ]  
}
```

[キ]へのヒント：「elseif側の条件式が
『生存 (ルールB)』を記述しているなら、
最初のif文は何を判定すべきだろうか？」

[ク]へのヒント：「『生存』した場合、次世
代の盤面 (m[i]) には何を書き込むべき
か？」

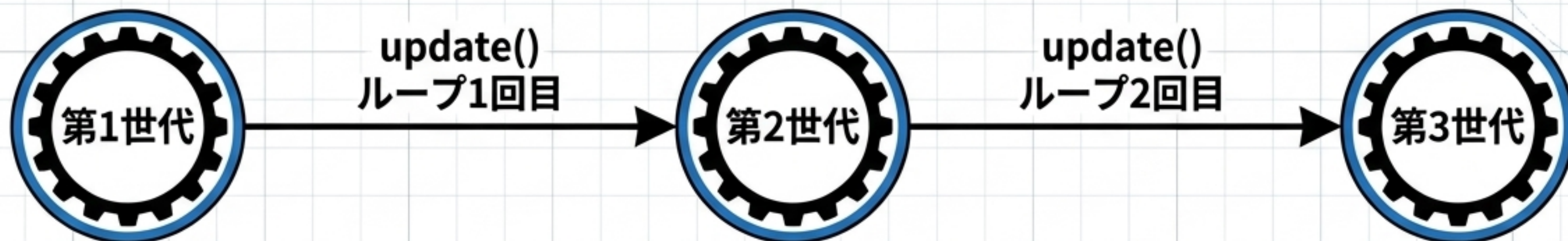
$i-1$ が N で
割り切れるマス

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	45	46	47	38	39	40
41	52	53	44	55	56	57	58	59	50
61	62	63	64	65	66	67	68	69	60
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

i が N で
割り切れるマス

Q: 「これらのハイライトされたマスは、盤面全体から見て『どこ』に位置していると言えるか？ 30字以内の日本語で表現してみよう。」

シナリオ：「 $p=2$ （第2世代）をテストしたい場合」



シナリオ：「 $p=2$ （第2世代）をテストしたい場合」

- ・ 図の通り、第1世代から第2世代へ移行するには、`update()` は「何回」必要か？
- ・ しかし、現在のプログラム `for (i を 1 から p まで)` は、 $p=2$ のとき「何回」回ってしまうか？

Q: 「目的の世代 (p) に到達するために、ループが回りすぎている。
開始値または終了値をどう調整すれば、回数を1回減らせるだろうか？」

設計図は完成した。

ルールの視覚化、2次元から1次元への変換、
コードとロジックの結びつき、
そして境界条件とデバッグ。
すべてのパーツの役割は明確になりました。

さあ、元の問題文に戻り、あなた自身の力で
解答欄を埋めてみましょう。