## Dojo Contours

**travail sur** : manipulation de tableaux, double boucle for

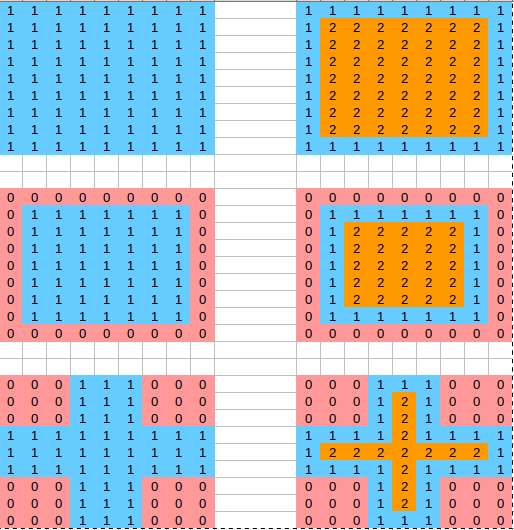
Énoncé : Remplir une zone en 2 dimensions en laissant une marge.

En entrée, un tableau à 2 dimensions contenant des 0 et des 1. Les 0 représentent une zone qu’il faut ignorer. Les 1 représentent la zone géométrique à remplir avec des 2. Lors du remplissage, il faut toujours laisser une marge de 1.

En sortie, un tableau à 2 dimensions avec la zone géométrique identifiée par des 1 remplie de 2 avec une marge de 1.

Exemples :

En entrée En sortie



Correction :

**public static function** action($input)

{

$result = **array**();

**for**($y = 0; $y < count($input); $y++){

$row = $input[$y];

**for** ($x = 0; $x < count($row); $x++){

$result[$y][$x] = $input[$y][$x];

**if** (1 === $input[$y][$x]){

**if** (**isset**($input[$y][$x-1]) &&

**isset**($input[$y][$x+1]) &&

**isset**($input[$y-1][$x]) &&

**isset**($input[$y+1][$x])){

$ok2Fill = **true**;

**for** ($i = $x - 1; $i <= $x + 1; $i++){

**for** ($j = $y - 1; $j <= $y + 1; $j++){

**if** ($input[$j][$i] !== 1 ){

$ok2Fill = **false**;

**break** 2;

}

}

}

**if** (**true** === $ok2Fill){

$result[$y][$x] = 2;

}

}

}

}

}

**return** $result;

}

Tests :

**public** $input55 = **array** (

**array**(1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1),

);

**public** $output55 = **array** (

**array**(1, 1, 1, 1, 1),

**array**(1, 2, 2, 2, 1),

**array**(1, 2, 2, 2, 1),

**array**(1, 2, 2, 2, 1),

**array**(1, 2, 2, 2, 1),

**array**(1, 1, 1, 1, 1),

);

**public** $input35 = **array** (

**array**(1, 1, 1),

**array**(1, 1, 1),

**array**(1, 1, 1),

**array**(1, 1, 1),

**array**(1, 1, 1),

**array**(1, 1, 1),

);

**public** $output35 = **array** (

**array**(1, 1, 1),

**array**(1, 2, 1),

**array**(1, 2, 1),

**array**(1, 2, 1),

**array**(1, 2, 1),

**array**(1, 1, 1),

);

**public** $inputL = **array** (

**array**(1, 1, 1, 1, 0, 0, 0, 0, 0),

**array**(1, 1, 1, 1, 0, 0, 0, 0, 0),

**array**(1, 1, 1, 1, 0, 0, 0, 0, 0),

**array**(1, 1, 1, 1, 0, 0, 0, 0, 0),

**array**(1, 1, 1, 1, 0, 0, 0, 0, 0),

**array**(1, 1, 1, 1, 1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1, 1, 1, 1, 1),

);

**public** $outputL = **array** (

**array**(1, 1, 1, 1, 0, 0, 0, 0, 0),

**array**(1, 2, 2, 1, 0, 0, 0, 0, 0),

**array**(1, 2, 2, 1, 0, 0, 0, 0, 0),

**array**(1, 2, 2, 1, 0, 0, 0, 0, 0),

**array**(1, 2, 2, 1, 0, 0, 0, 0, 0),

**array**(1, 2, 2, 1, 1, 1, 1, 1, 1),

**array**(1, 2, 2, 2, 2, 2, 2, 2, 1),

**array**(1, 2, 2, 2, 2, 2, 2, 2, 1),

**array**(1, 1, 1, 1, 1, 1, 1, 1, 1),

);

**public** $inputCross = **array** (

**array**(0, 0, 0, 1, 1, 1, 0, 0, 0),

**array**(0, 0, 0, 1, 1, 1, 0, 0, 0),

**array**(0, 0, 0, 1, 1, 1, 0, 0, 0),

**array**(1, 1, 1, 1, 1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1, 1, 1, 1, 1),

**array**(1, 1, 1, 1, 1, 1, 1, 1, 1),

**array**(0, 0, 0, 1, 1, 1, 0, 0, 0),

**array**(0, 0, 0, 1, 1, 1, 0, 0, 0),

**array**(0, 0, 0, 1, 1, 1, 0, 0, 0),

);

**public** $outputCross = **array** (

**array**(0, 0, 0, 1, 1, 1, 0, 0, 0),

**array**(0, 0, 0, 1, 2, 1, 0, 0, 0),

**array**(0, 0, 0, 1, 2, 1, 0, 0, 0),

**array**(1, 1, 1, 1, 2, 1, 1, 1, 1),

**array**(1, 2, 2, 2, 2, 2, 2, 2, 1),

**array**(1, 1, 1, 1, 2, 1, 1, 1, 1),

**array**(0, 0, 0, 1, 2, 1, 0, 0, 0),

**array**(0, 0, 0, 1, 2, 1, 0, 0, 0),

**array**(0, 0, 0, 1, 1, 1, 0, 0, 0),

);

**public function** test1()

{

$this->assertEquals($this->output55, \wcs\KataExemple::*action*($this->input55));

$this->assertEquals($this->output35, \wcs\KataExemple::*action*($this->input35));

$this->assertEquals($this->outputL, \wcs\KataExemple::*action*($this->inputL));

$this->assertEquals($this->outputCross, \wcs\KataExemple::*action*($this->inputCross));

}