COMP1511 PROGRAMMING FUNDAMENTALS

LECTURE 7

An array of arrays, 2D

LAST WEEK...

IN WEEK 3...

- Talked about the importance of style - work neatly as you go!
- Discovered functions (separate chunks of code for reuse, help to segment the problem)
- Got introduced to arrays - homogenous collections - stores the same type of variable in a collection

THIS LECTURE...

TODAY...

- Assignment 1 kick off
- Recap basic arrays
- Array of arrays
- Array of structs

WHERE IS THE CODE?

Live lecture code can be found here:

HTTPS://CGI.CSE.UNSW.EDU.AU/~CS1511/22T1/LIVE/WEEK04/

ASSIGNMENT 1

RELEASED TODAY

- Assignment 1 has been released
- CS Explorer - explore the far flung corners of the CSE Building whilst dealing with monsters
- Aims of the assignment
  - Apply arrays and two-dimensional arrays in solving problems
  - Apply good style to your code
  - Apply the use of functions in code
  - Practice skills in debugging code, and skills in patience as you search for one missing semi-colon

ASSIGNMENT 1

LIVESTREAM

- The Assignment has 4 stages, each stage ramps up with difficulty (just like the lab exercises)
- Suggest going through the stages chronologically - do not skip stages
- Live Stream to go through the assignment in more detail:
  - Wednesday 3:00pm
  - Link for the livestream: https://youtu.be/xAP8dWxBXP8

RECAP OF ARRAYS

Remember that arrays:

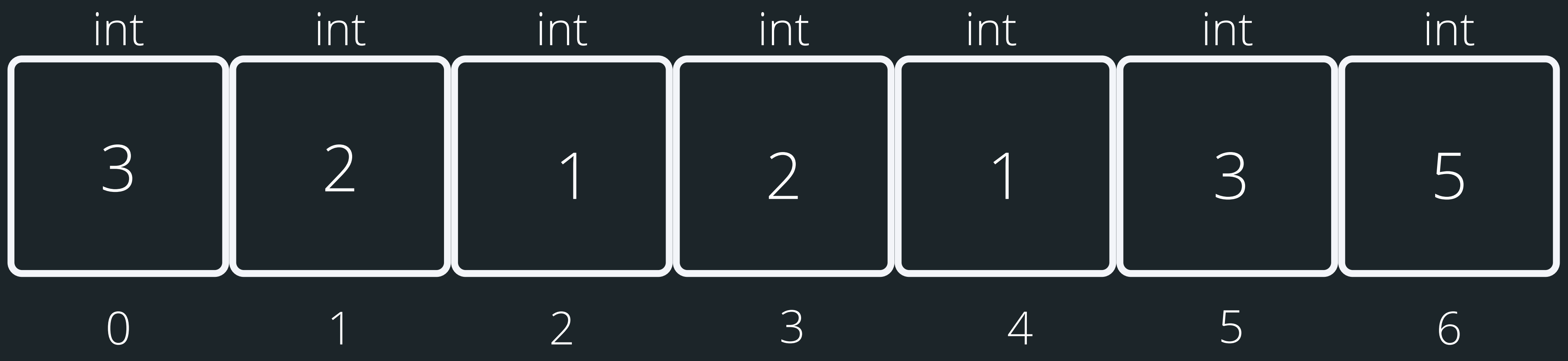
- are a collection all of the same type
- are declared by using a type, name and a size of the array
- you can easily access individual elements of an array by using an index
- Indexing starts at 0 and moves through until (size - 1) of the array
- go hand in hand with while loops that make it easy to work through an array

RECAP OF ARRAYS

- So let's say we have this declared and initialised:

```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};
```

- This is what it looks like visually:



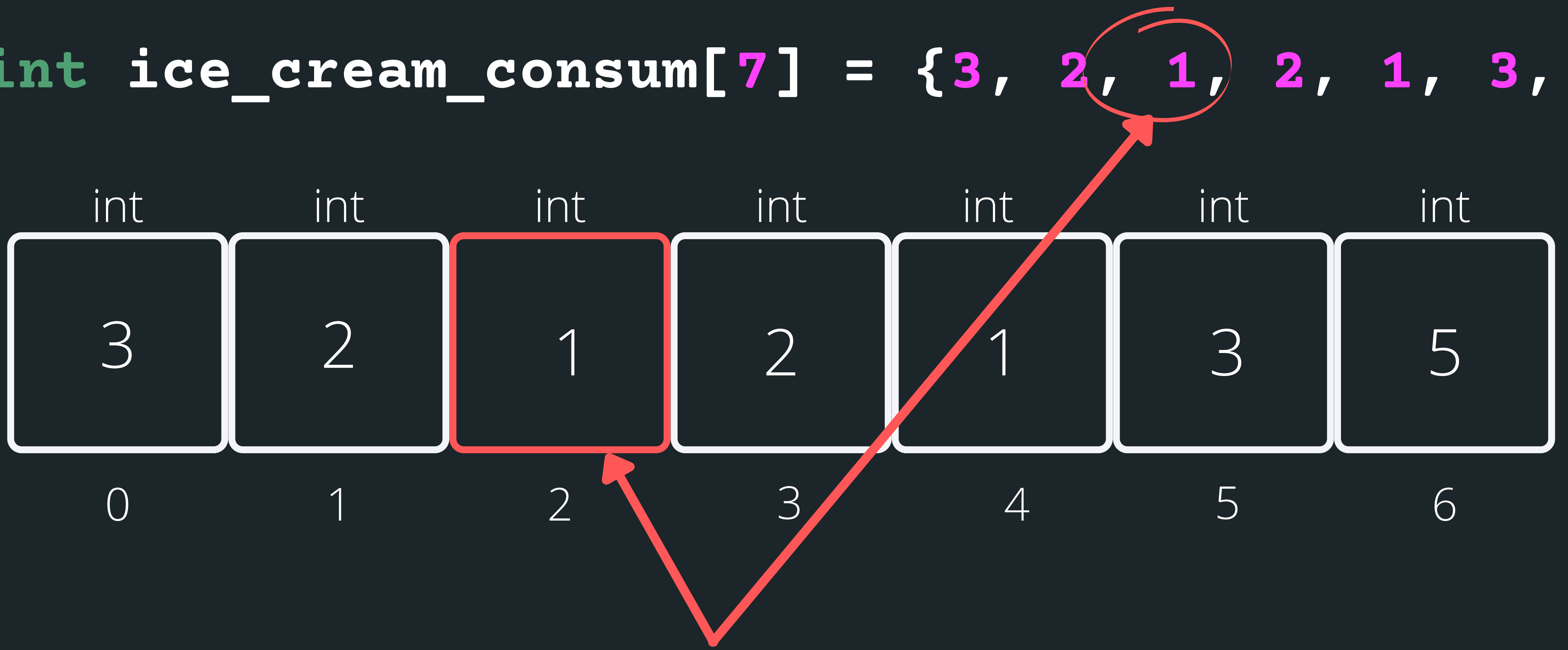
this array holds 7 integers

Note that indexing starts at 0

RECAP OF ARRAYS

- You can access any element of the array by referencing its index
- Note, that indexes start from 0
- Trying to access an index that does not exist, will result in an error

```
int ice_cream_consum[7] = {3, 2, 1, 2, 1, 3, 5};
```



If I wanted the third element of the array

The index would be 2, so to access it:

ice_cream_consum[2]

RECAP OF ARRAYS

AN EXAMPLE PROBLEM

Problem: A user is asked to enter 10 numbers. We will then go through these numbers and find the lowest number and output what the lowest number is to the user.

`lowest_number.c`

YOU CAN HAVE AN ARRAY OF ANYTHING

AN ARRAY OF STRUCTS

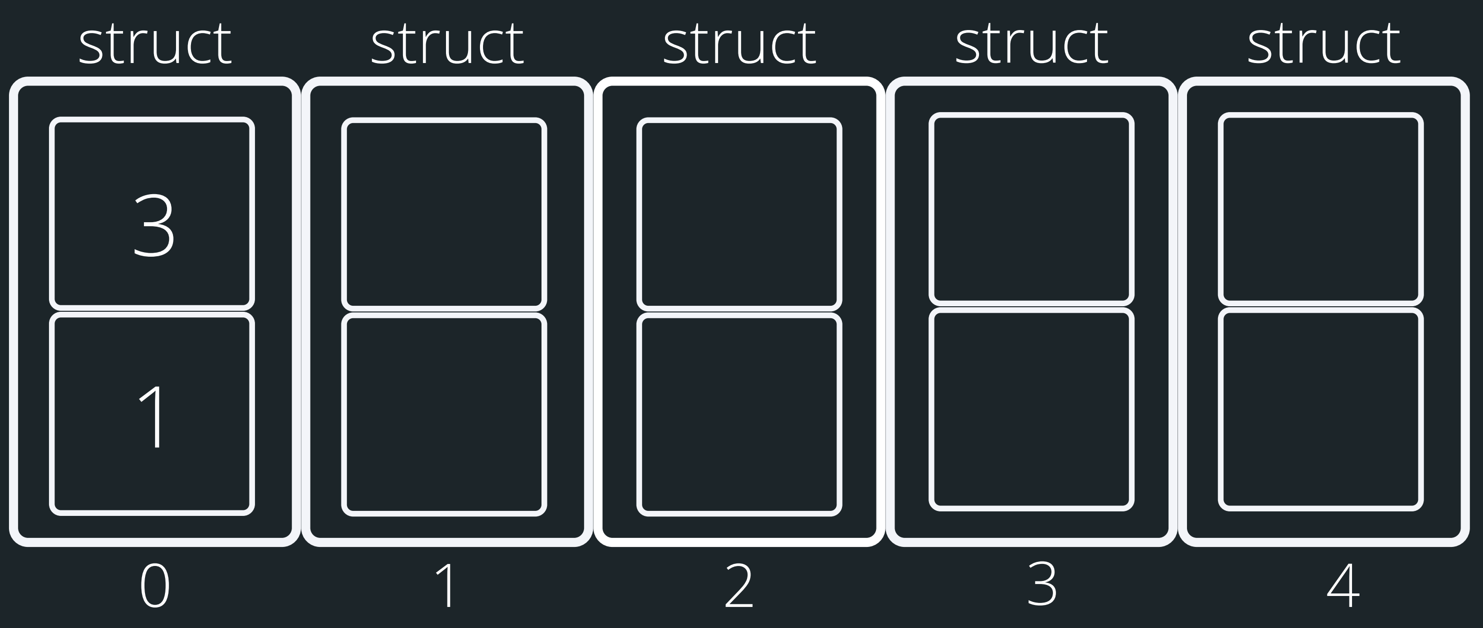
The struct for a coordinate point:

```
struct coordinate {
    int x;
    int y;
};
```

An array of structs declared:

```
struct coordinate map[5];
```

An array of structs visually:



```
map[0].x = 3;
map[0].y = 1;
```

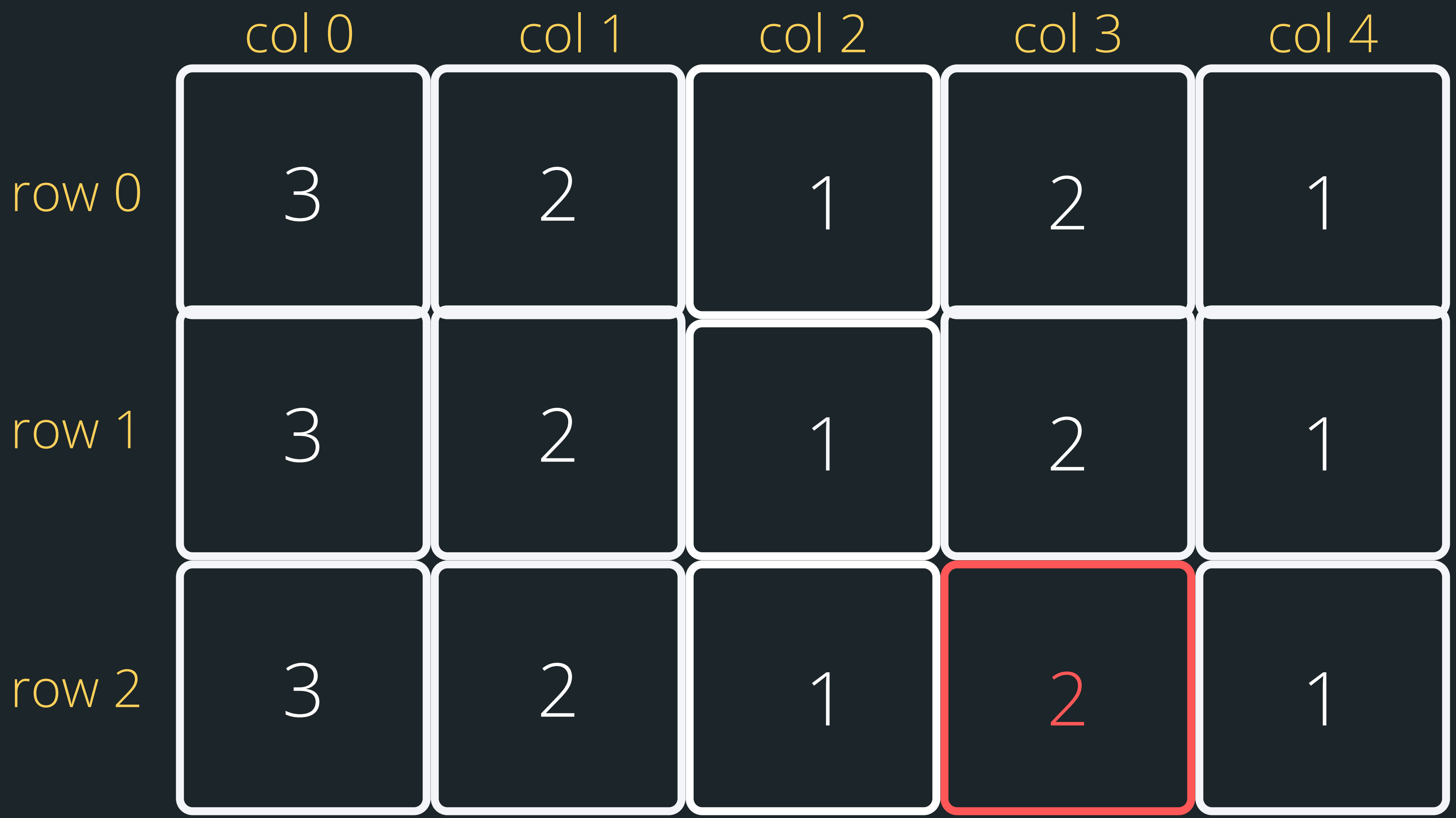
ACCESSING AN ELEMENT INSIDE ARRAY OF ARRAYS

An array of arrays is basically a grid. To declare an array of arrays:

```
type array_name[num of rows][num of columns];
int array[3][5];
```

To access an element now you will need to:

```
array[2][3];
```

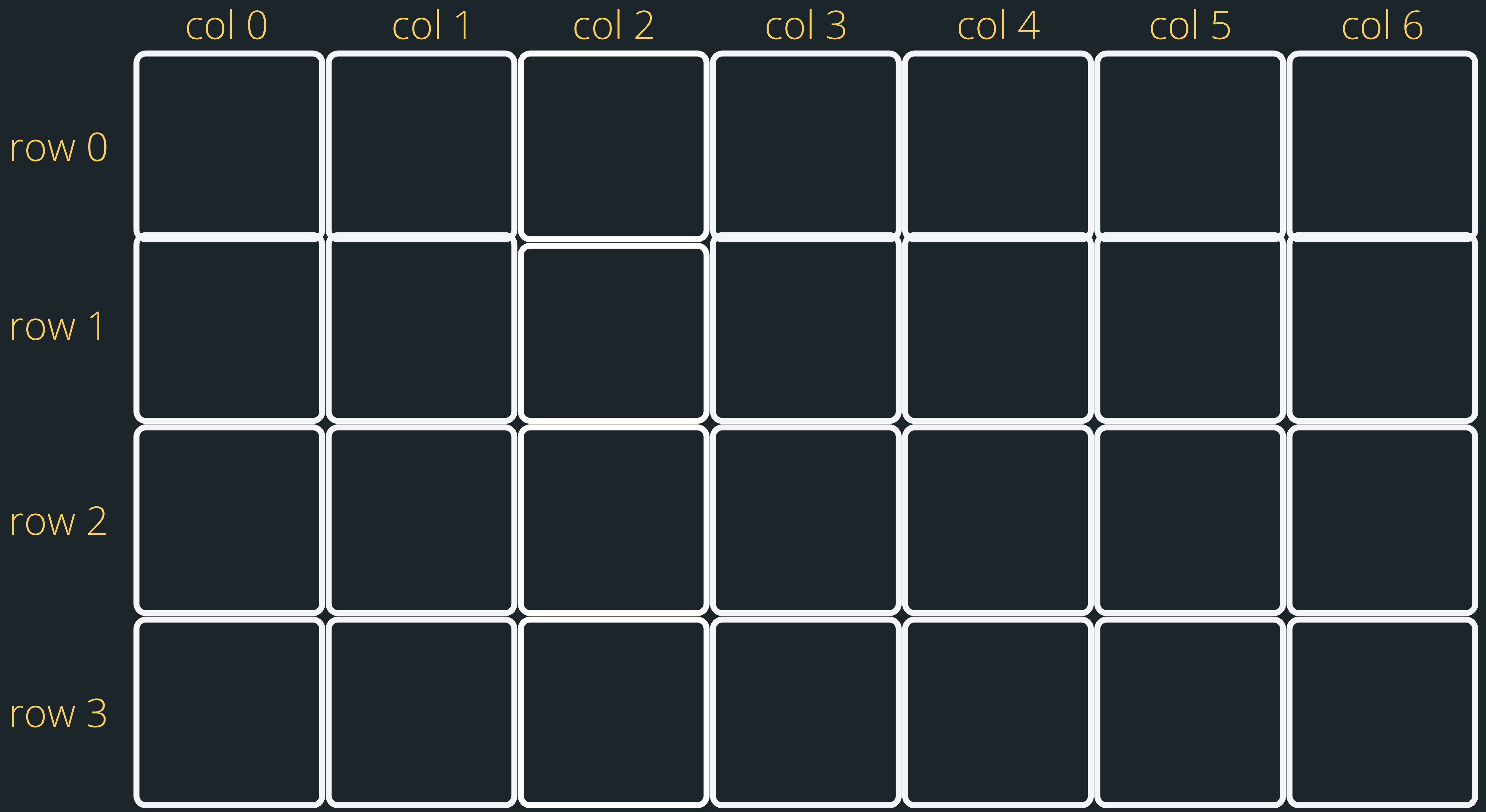


	col 0	col 1	col 2	col 3	col 4
row 0	3	2	1	2	1
row 1	3	2	1	2	1
row 2	3	2	1	2	1

ARRAY OF ARRAYS

Think of the problem last week where we tracked ice-cream consumption for a week. What if I want to do this for a month (a week at a time)?

```
int ice_cream[4][7];
```



REMEMBER A WHILE LOOP INSIDE A WHILE LOOP TO PRINT A GRID?

Do you remember when we printed out a grid of numbers in Week 2 (Friday night vibes)?

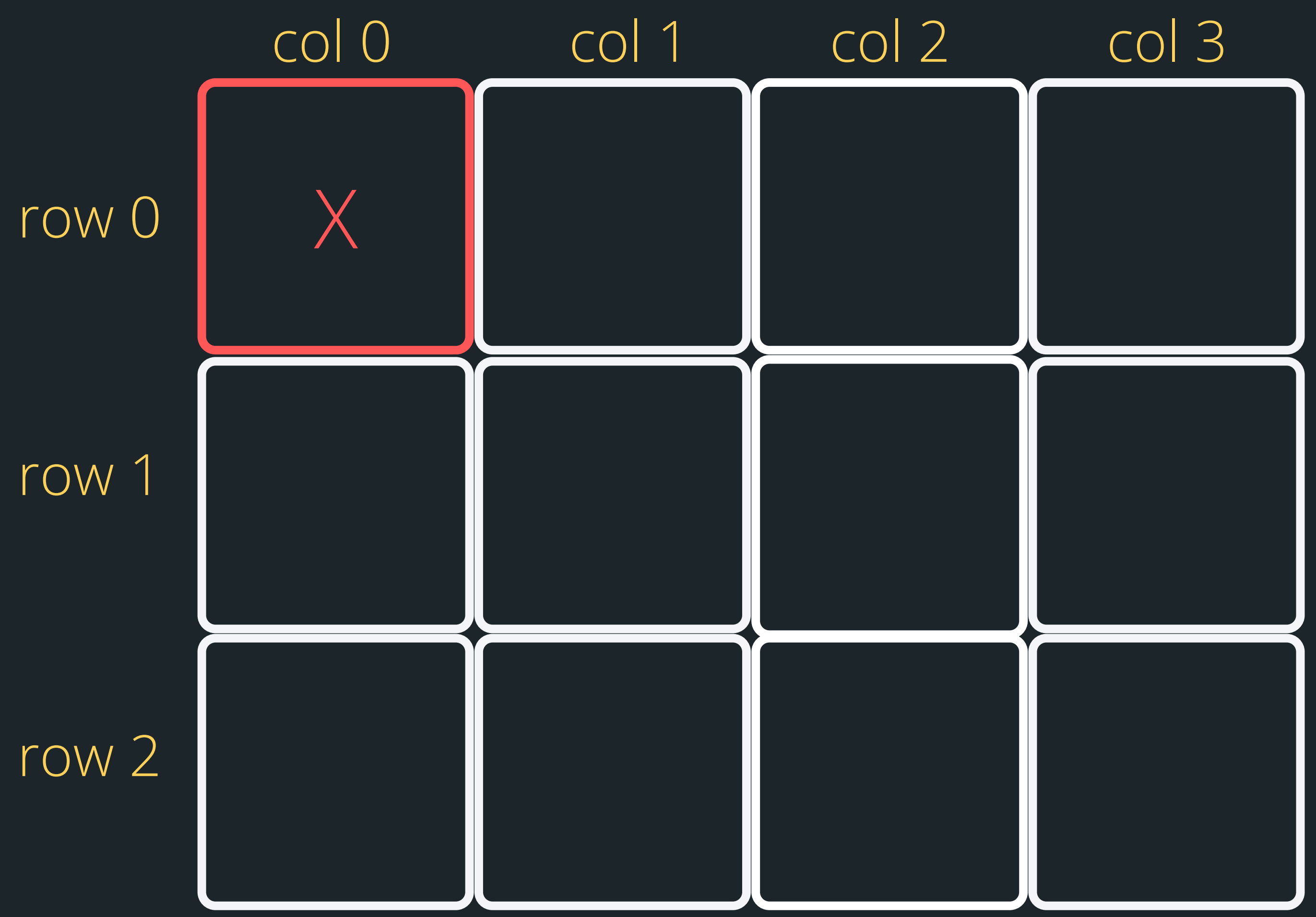
```
int row = 0;
while (row <= SIZE){
    int col = 0;
    while (col <= SIZE){
        printf("%d", col);
        col++;
    }
printf("\n");
row++;
}
```

How can we transfer this knowledge to print out an array of arrays?

TRANSFER THIS TO AN ARRAY:

FIRST RUN AROUND THE SUN:
OUTSIDE WHILE
ROW = 0
INSIDE WHILE
COL = 0

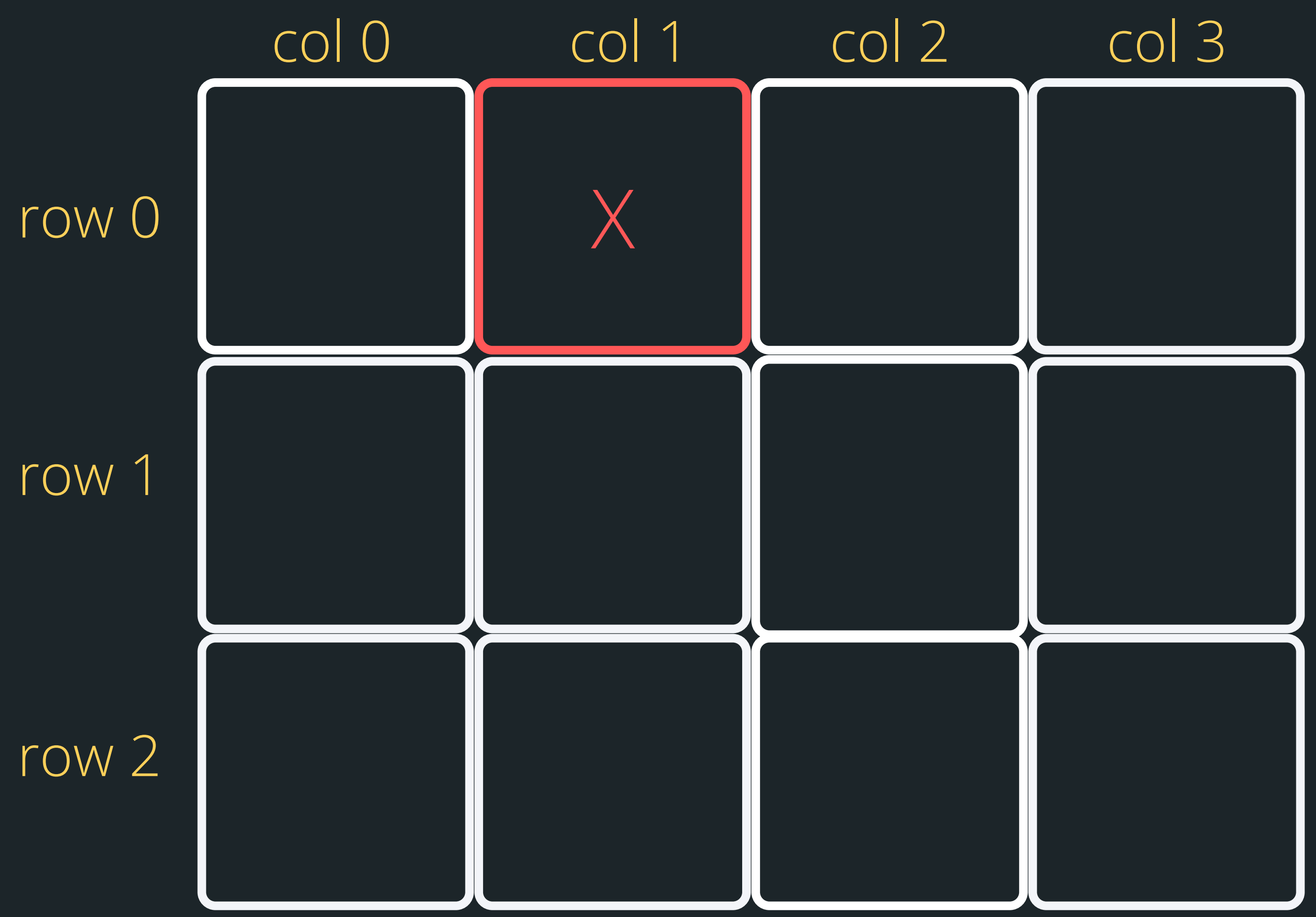
```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

FIRST RUN AROUND THE SUN:
OUTSIDE WHILE
ROW = 0
INSIDE WHILE
COL = 1

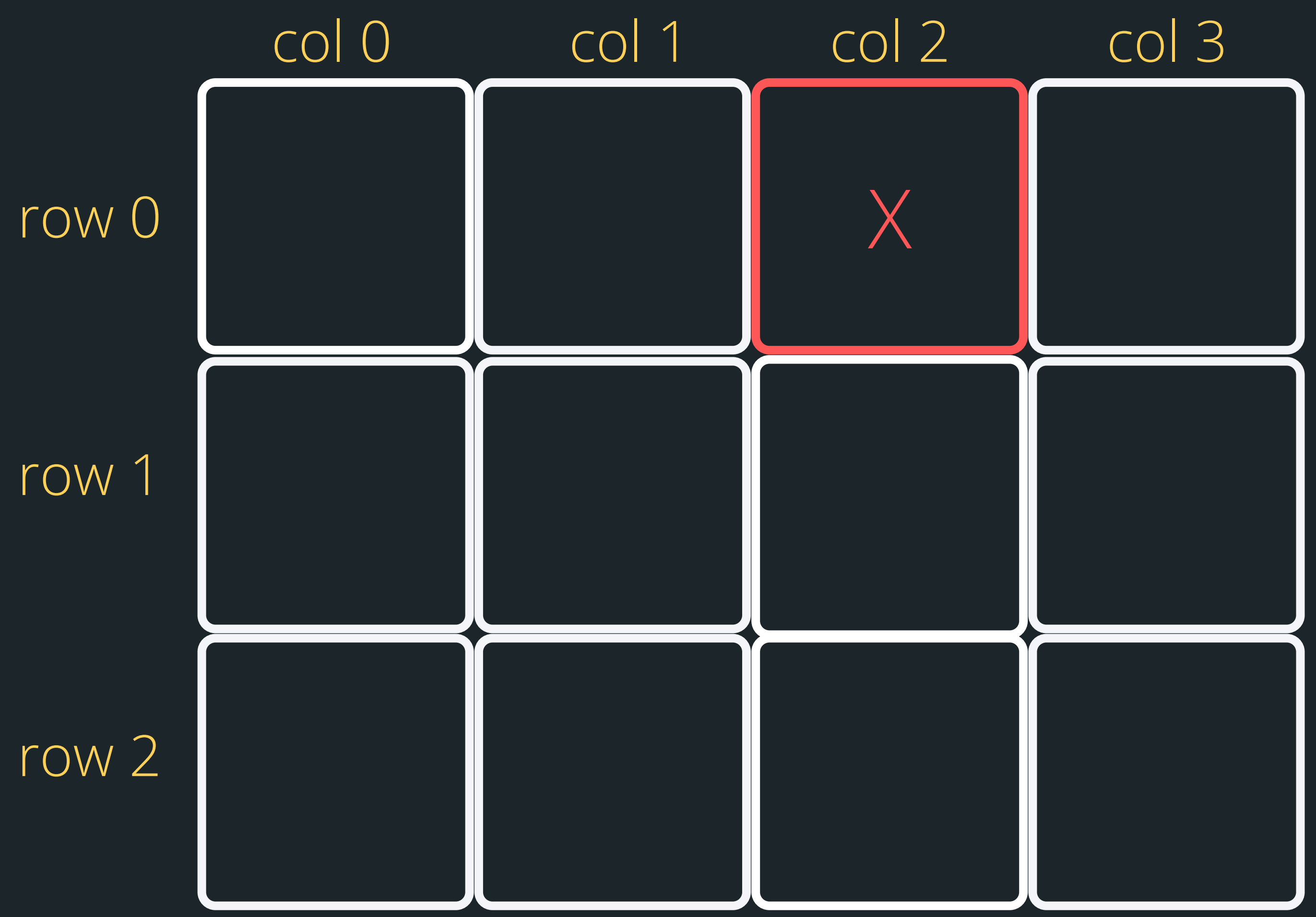
```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

FIRST RUN AROUND THE SUN:
OUTSIDE WHILE
    ROW = 0
INSIDE WHILE
    COL = 2

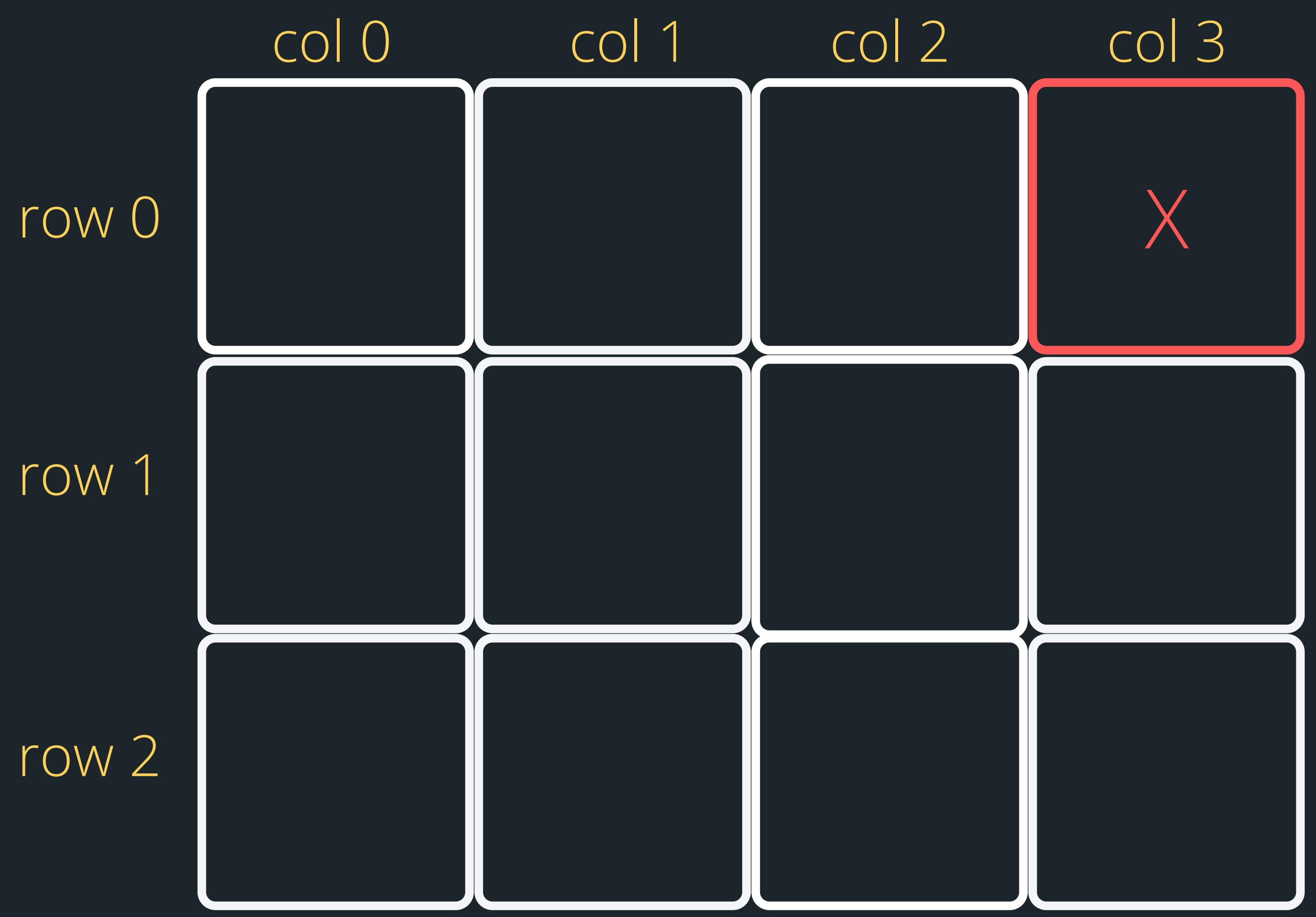
```c
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

FIRST RUN AROUND THE SUN:
OUTSIDE WHILE
    ROW = 0
INSIDE WHILE
    COL = 3

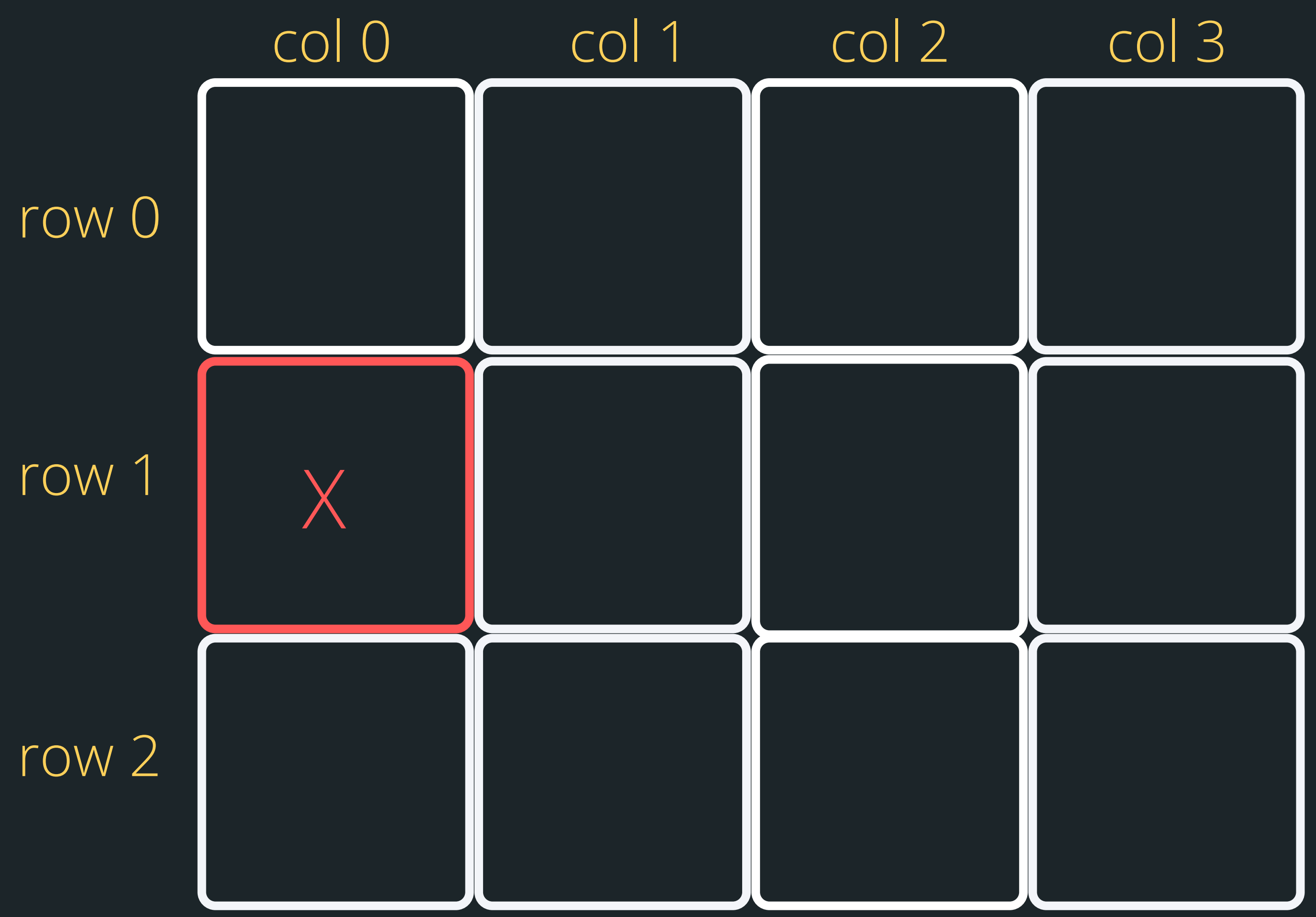
```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

SECOND RUN AROUND THE SUN:
OUTSIDE WHILE
    ROW = 1
INSIDE WHILE
    COL = 0

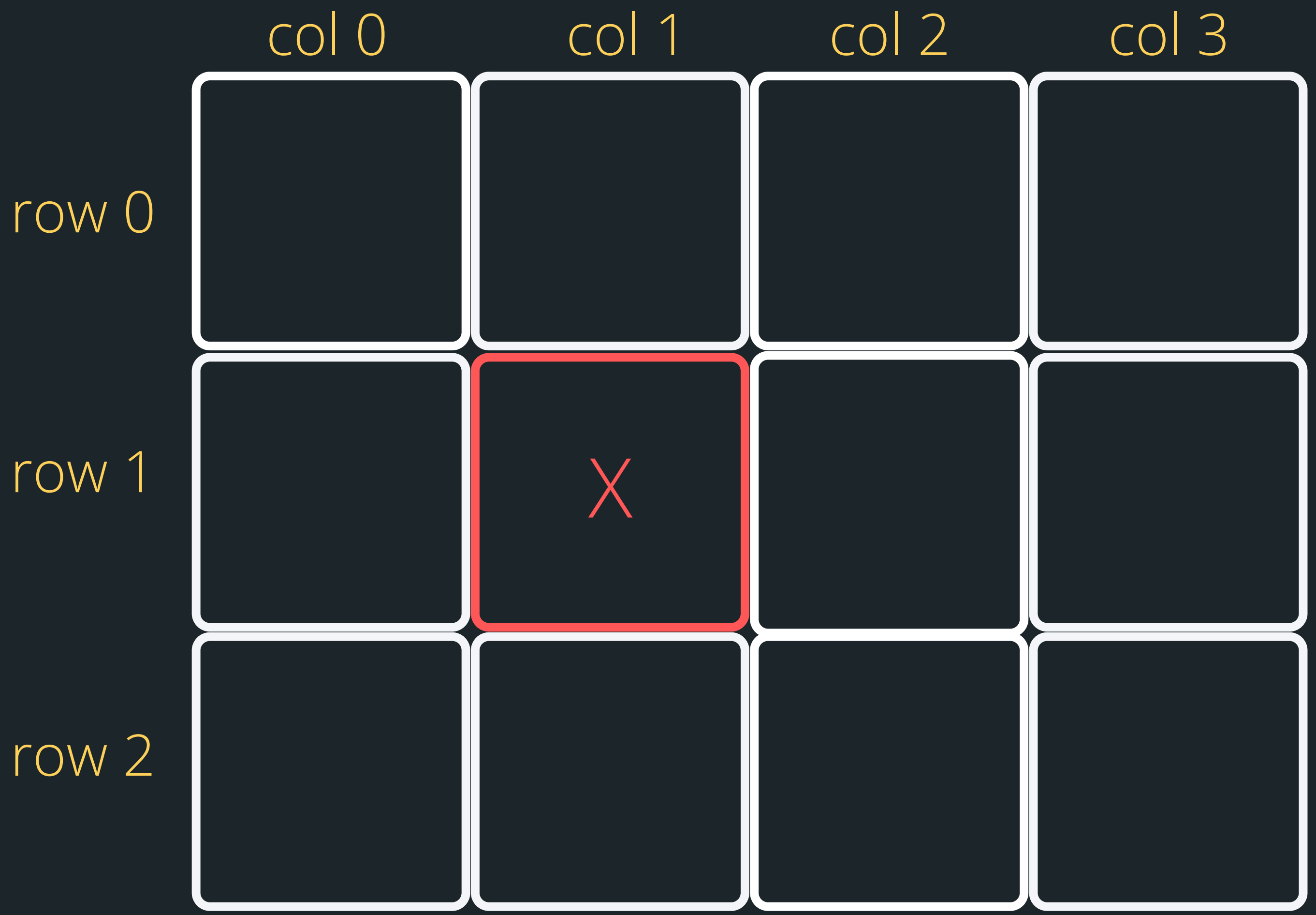
```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

SECOND RUN AROUND THE SUN:
OUTSIDE WHILE
ROW = 1
INSIDE WHILE
COL = 1

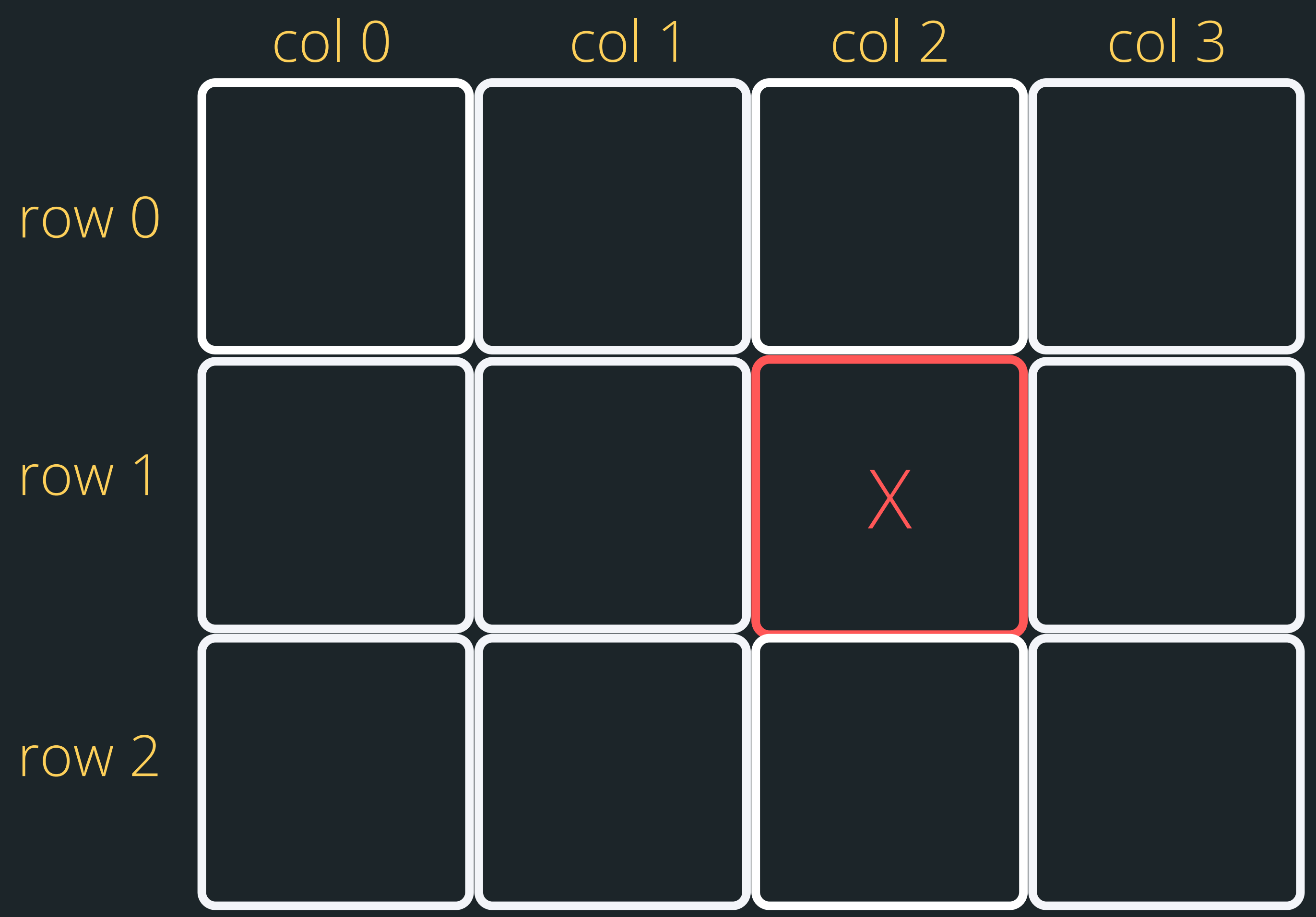
```
int array[3][4];
int row = 0;
while (row <= 3){
     int col = 0;
     while (col <= 4){
          printf("%d", array[row][col]);
          col++;
     }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

SECOND RUN AROUND THE SUN:
OUTSIDE WHILE
    ROW = 1
INSIDE WHILE
    COL = 2

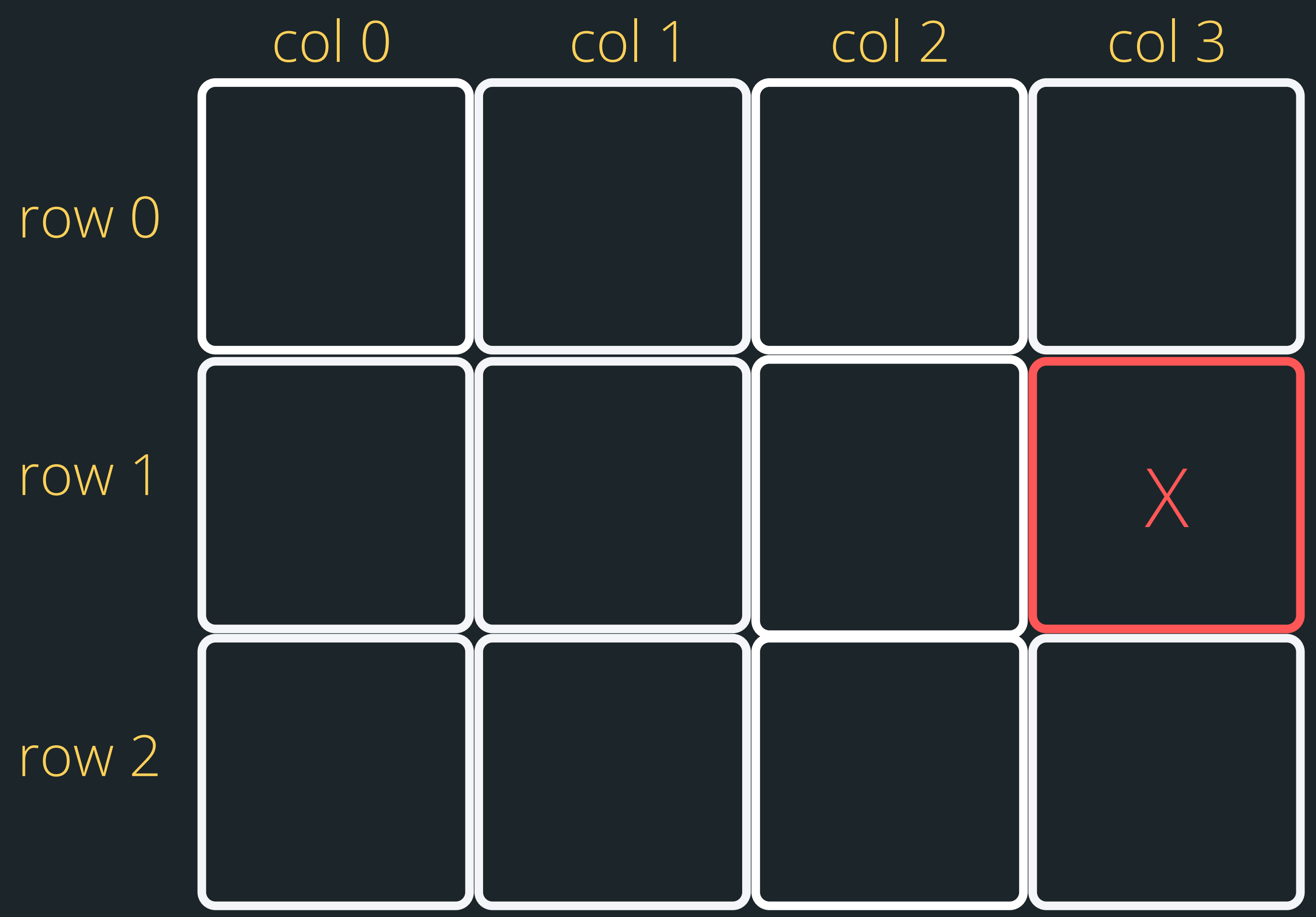
```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

SECOND RUN AROUND THE SUN:
OUTSIDE WHILE
ROW = 1
INSIDE WHILE
COL = 3

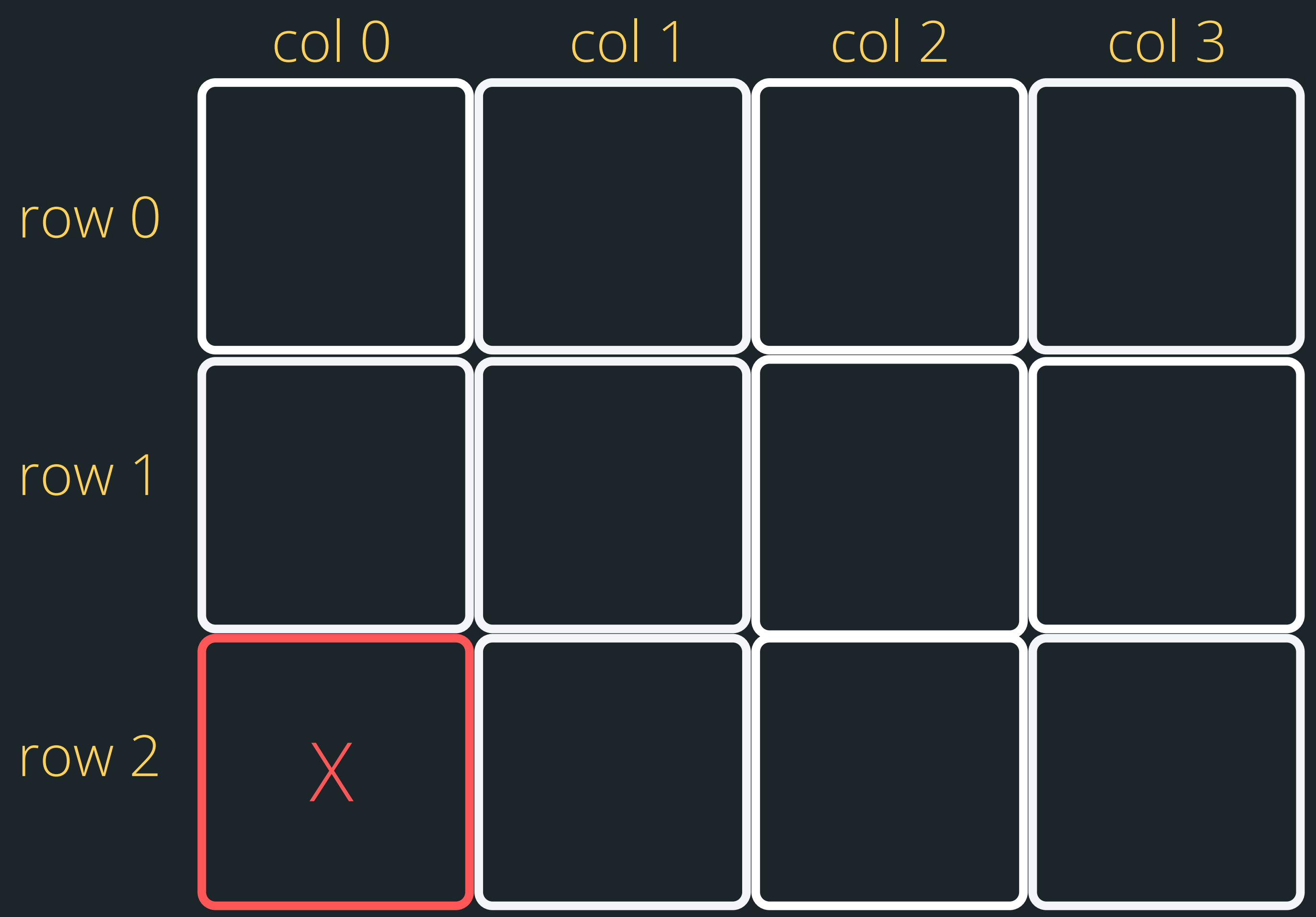
```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

THIRD RUN AROUND THE SUN:
OUTSIDE WHILE
    ROW = 2
INSIDE WHILE
    COL = 0

```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

THIRD RUN AROUND THE SUN:

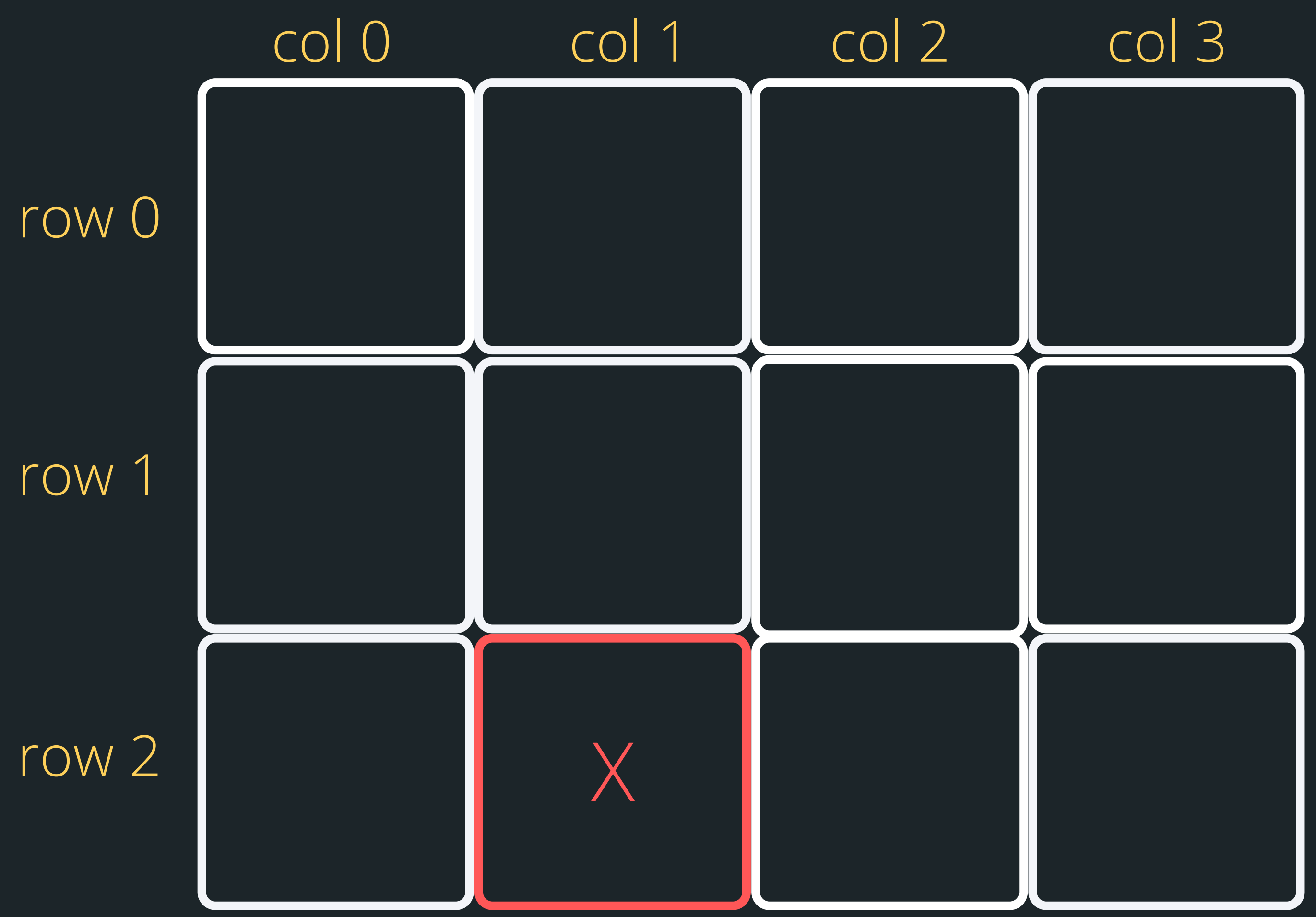
OUTSIDE WHILE

ROW = 2

INSIDE WHILE

COL = 1

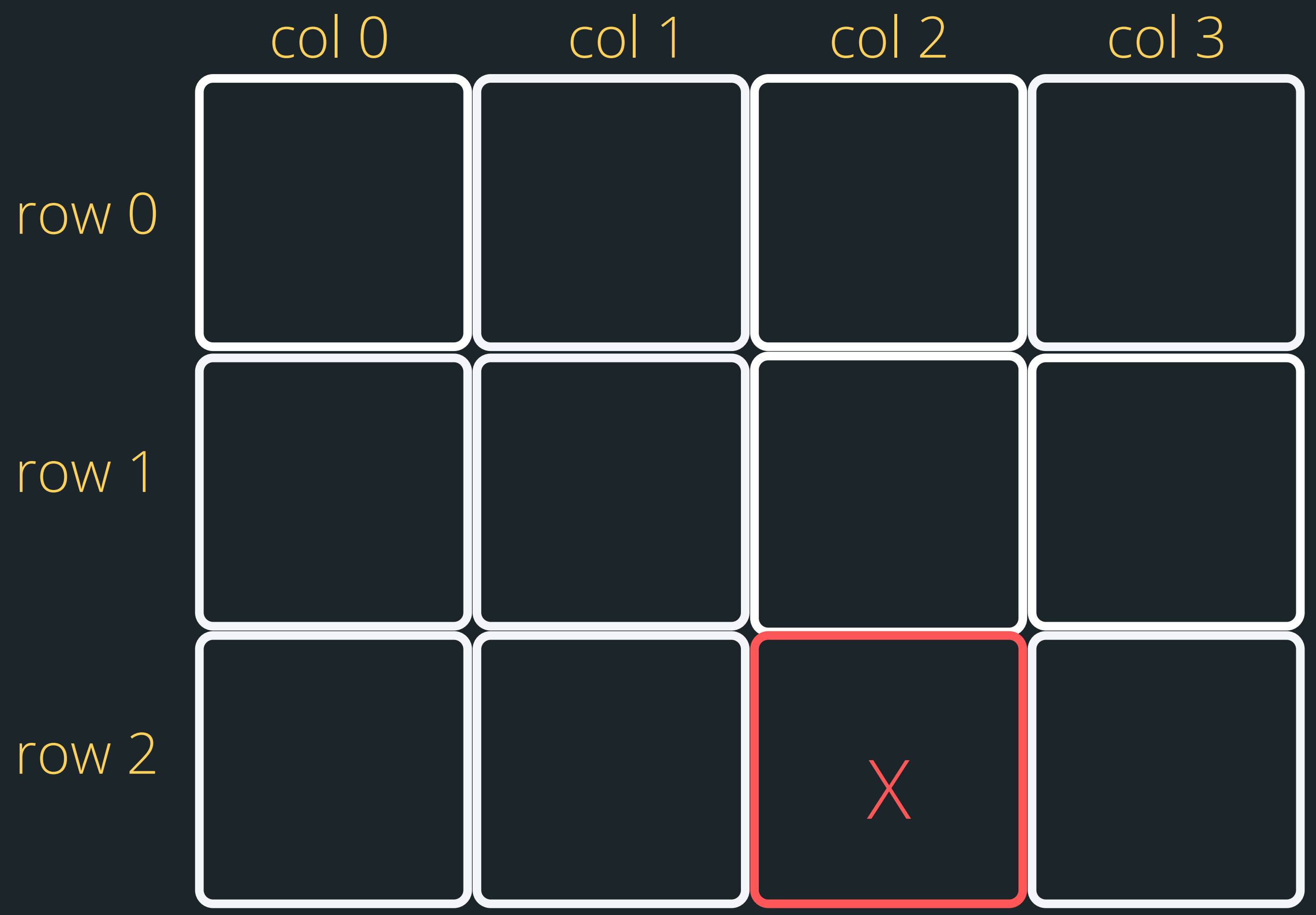
```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

THIRD RUN AROUND THE SUN:
OUTSIDE WHILE
ROW = 2
INSIDE WHILE
COL = 2

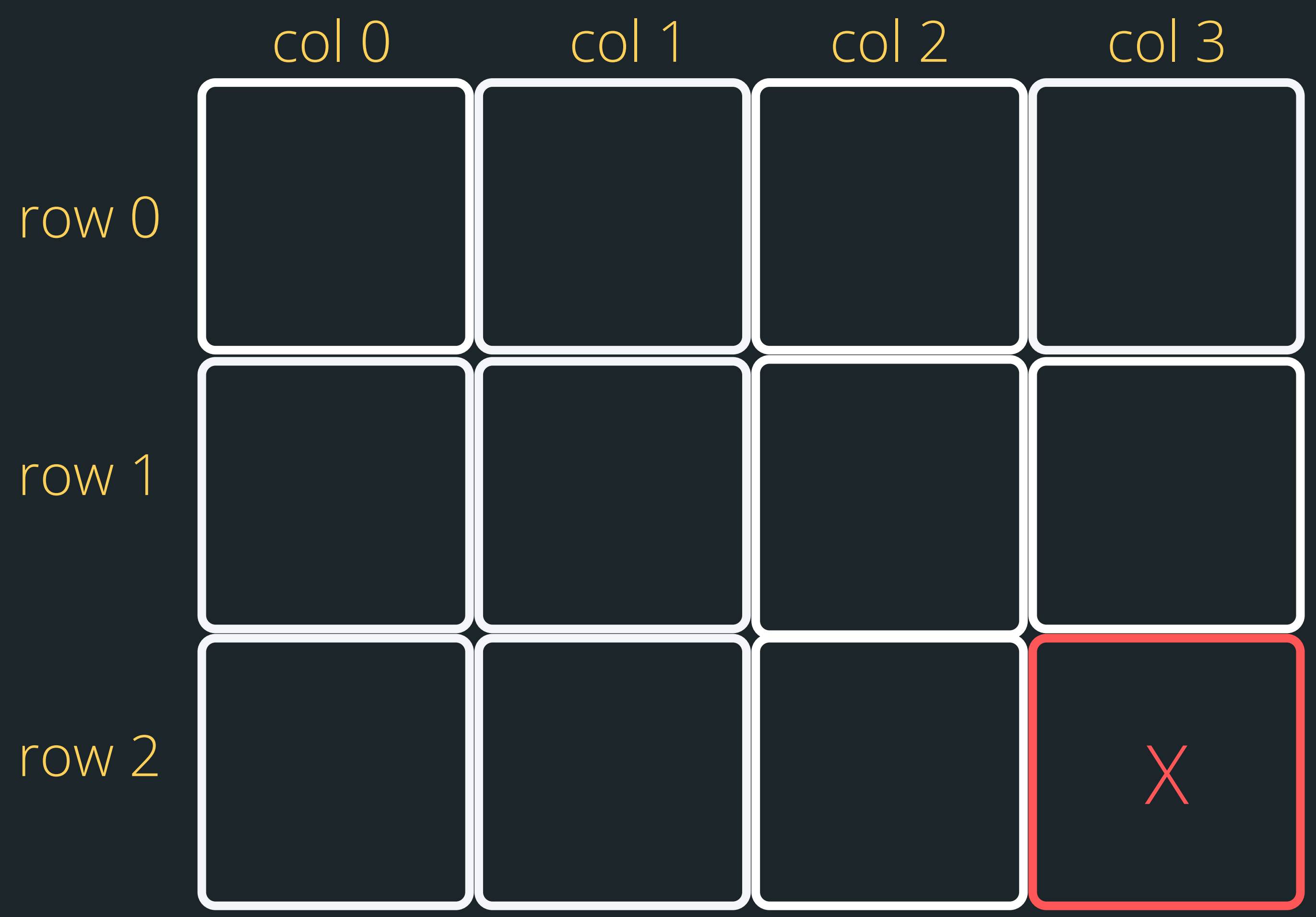
```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
         printf("%d", array[row][col]);
         col++;
    }
printf("\n");
row++;
}
```



TRANSFER THIS TO AN ARRAY:

THIRD RUN AROUND THE SUN:
OUTSIDE WHILE
ROW = 2
INSIDE WHILE
COL = 3

```
int array[3][4];
int row = 0;
while (row <= 3){
    int col = 0;
    while (col <= 4){
        printf("%d", array[row][col]);
        col++;
    }
printf("\n");
row++;
}
```



BREAK TIME

TIME TO STRETCH

There are five bags of gold that all look identical, and each has ten gold pieces in it. One of the five bags has fake gold in it. The real gold, fake gold, and all five bags are identical in every way, except the pieces of fake gold each weigh 1.1 grams, and the real gold pieces each weigh 1 gram. You have a perfectly accurate digital gram scale and can use it only once. How do you determine which bag has the fake gold?

HARDER PROBLEM

I am on the hunt for ice-cream (what else is new?). I I would like to explore a certain area in Kinsgsford to see if I can find ice-cream, on 10x10 grid. I am able to move around this section of Kingsford (left, right, up and down) and want to explore as many places in this section as possible, so as not to miss an ice-cream opportunity. Once I have explored the section in Kingsford, or I go to the same place more than once - it will be time to go home.

We can make this problem more complex if we have time with either Tom or Tammy deciding to join me on the hunt!

HARDER PROBLEM

- Like Assignment 1, we will have starter code
- The starter code provides us partial functionality for printing out our square park grid
- Let's move over to gedit and start solving this problem by breaking it up into components

`ice_cream_hunt.c`

Feedback please!

I value your feedback and use to pace the lectures and improve your overall learning experience. If you have any feedback from today's lecture, please follow the link below. Please remember to keep your feedback constructive, so I can action it and improve the learning experience.

https://www.menti.com/gq64ap2zzq

WHAT DID WE LEARN TODAY?

ASSIGNMENT 1 IS RELEASED

LIVESTREAM on Wednesday 3pm: https://youtu.be/xAP8dWxBXP8

RECAP 1D ARRAYS

lowest_number.c

AN ARRAY OR ARRAYS (2D)

print_2Darray.c

HARDER PROBLEM

ice_cream_hunt.c

REACH OUT

CONTENT RELATED QUESTIONS

Check out the forum

ADMIN QUESTIONS

cs1511@cse.unsw.edu.au