C Programming

Initialising a 2-Dimensional (2-D) Array

Remember, to initialise a 1-D array, you do as follows:

```
int numbers[5] = {2,4,6,8,10};
```

Moving on, in order to also initialise a 2-D array, you can do as follows:

Let's assume we have the following 2-D array:

```
int numbers[4][3];
```

1. Method One

```
int numbers[4][3] = {1 ,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

2. Method Two

```
int numbers[4][3] = {   1, 2, 3,
                        4, 5, 6,
                        7, 8, 9,
                        10, 11, 12
                    };

int i, j;
```

3. Method Three

```
int numbers[4][3] = {   {1, 2, 3} ,
                        {4, 5, 6} ,
                        {7, 8, 9} ,
                        {10, 11, 12}
                    };
```

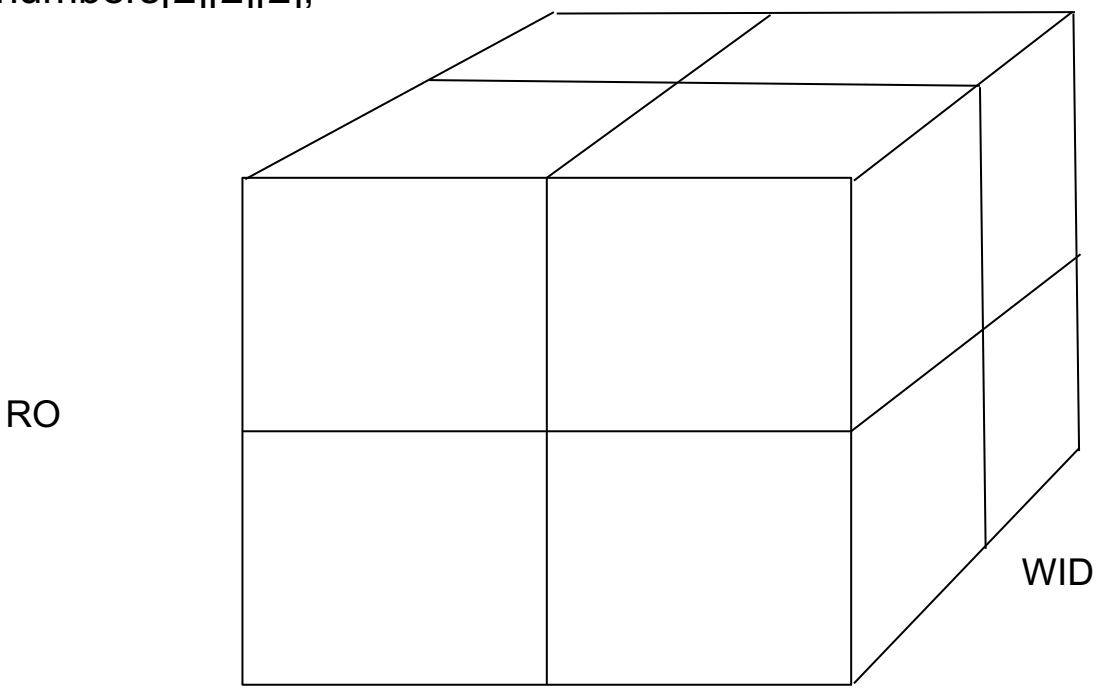
Repl 8.1: https://replit.com/@michaelTUDublin/81-Initialising-a-2-D-array#main.c

Multi-dimensional arrays (3-D, 4-D, etc.,)

In industry, there are sectors where multi-dimensional arrays are commonly used, 3-D, 4-D arrays, etc., are used in software development

Here is a visualisation of what a 3-D array would look

int numbers[2][2][2];



Programming Pitfalls

1. Be careful when using Symbolic names

e.g.,

```
#define SIZE 10;  // Invalid. There is no semi-colon at the end

#define SIZE = 10  // Invalid. You do not use an equals operator here

#define SIZE 10   // Valid and correct
```

2. Make sure you use square brackets [ ] and not curly brackets { } for specifying the size of an array