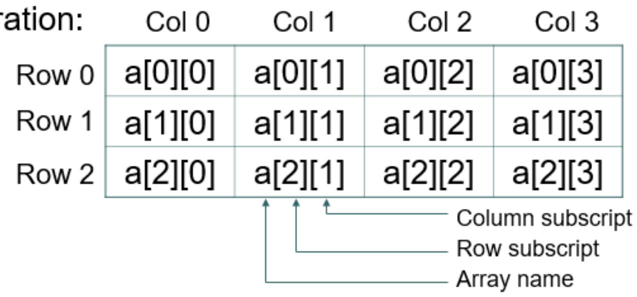
Two Dimensional (2D) / Double – Subscripted Array

The two dimensional or double-subscripted array can be represented as a table with rows and columns. In other words, a double-subscripted array is used when data is represented using a table. It requires two subscripts to identify a particular element. By convention, the first subscript identifies the element's row and the second identifies the element's column.

Illustration:



	Col 0	Col 1	Col 2	Col 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Syntax for the declaration:

dataType arrayName[rowSize][rowColumn];

For example, to declare a 2D integer array `table` of three rows and 4 columns:

```
int table[3][4];
```

To illustrate:

	0	1	2	3
0				
1				
2				

Initializing a 2D Array

Here are examples on how to initialize 2D arrays:

1. `int table[3][4]={0};`

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

Similar to a single-subscripted array, the initializer assigns the elements to its proper location. If lacking, the rest is initialized to 0. Thus, 0 is assigned to index [0][0] and the rest until index [2][3] is assigned 0.

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

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

The elements are assigned per row. If the row is full, assignment proceeds to the next row and so on.

```
3. int table[3][4] = {1,2,3,4,5,6,7,8,9,10};
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	0	0

Similar to the first example, if the initializer list has lesser elements, the remaining elements will be 0.

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

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

Initializer list can also be grouped per row.

```
5. int table[3][4] = {{1,2,3},{5,6},{9,10,11,12}};
```

	0	1	2	3
0	1	2	3	0
1	5	6	0	0
2	9	10	11	12

Since the initializer list is grouped per row, if a row lacks an element, 0 will be assigned to the remaining elements in that row.

Accessing Elements of the 2D Array

1. By the individual elements

For example,

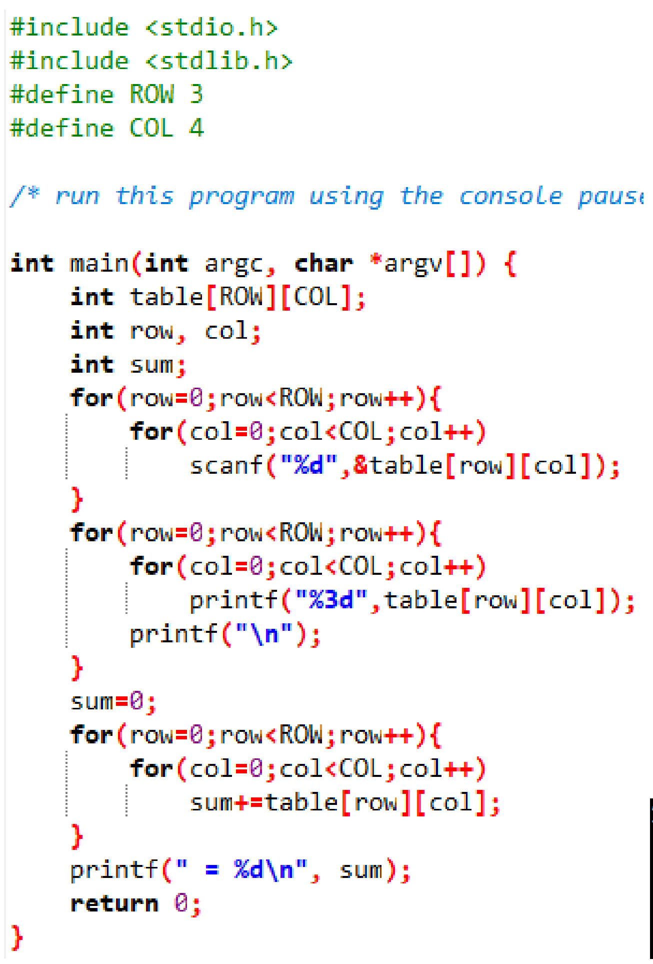
```
table[0][0] = 6;
printf(“%d”,table[2][1]);
```

2. Using a nested-loop

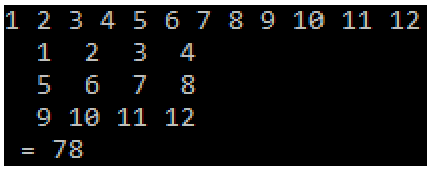
```
for(row=0;row<3; row++){

   for(col=0;col<4;col++)

         printf(“%d”,table[row][col]);

}
```

By convention, in accessing elements in a 2D array, nested-loop is used. The outer loop is the row counter while the inner loop is the column counter.

Here is a sample program with the output:



```
#include <stdio.h>
#include <stdlib.h>
#define ROW 3
#define COL 4

/* run this program using the console paus

int main(int argc, char *argv[]) {
    int table[ROW][COL];
    int row, col;
    int sum;
    for(row=0;row<ROW;row++){
        for(col=0;col<COL;col++)
            scanf("%d",&table[row][col]);
    }
    for(row=0;row<ROW;row++){
        for(col=0;col<COL;col++)
            printf("%3d",table[row][col]);
        printf("\n");
    }
    sum=0;
    for(row=0;row<ROW;row++){
        for(col=0;col<COL;col++)
            sum+=table[row][col];
    }
    printf(" = %d\n", sum);
    return 0;
}
```

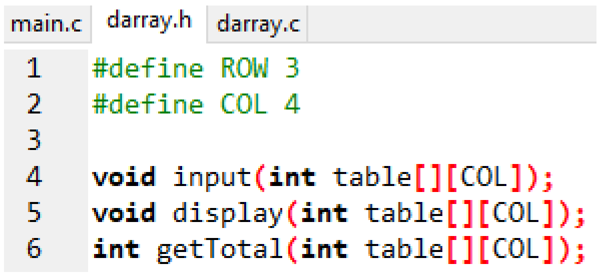


```
1 2 3 4 5 6 7 8 9 10 11 12
  1  2  3  4
  5  6  7  8
  9 10 11 12
 = 78
```

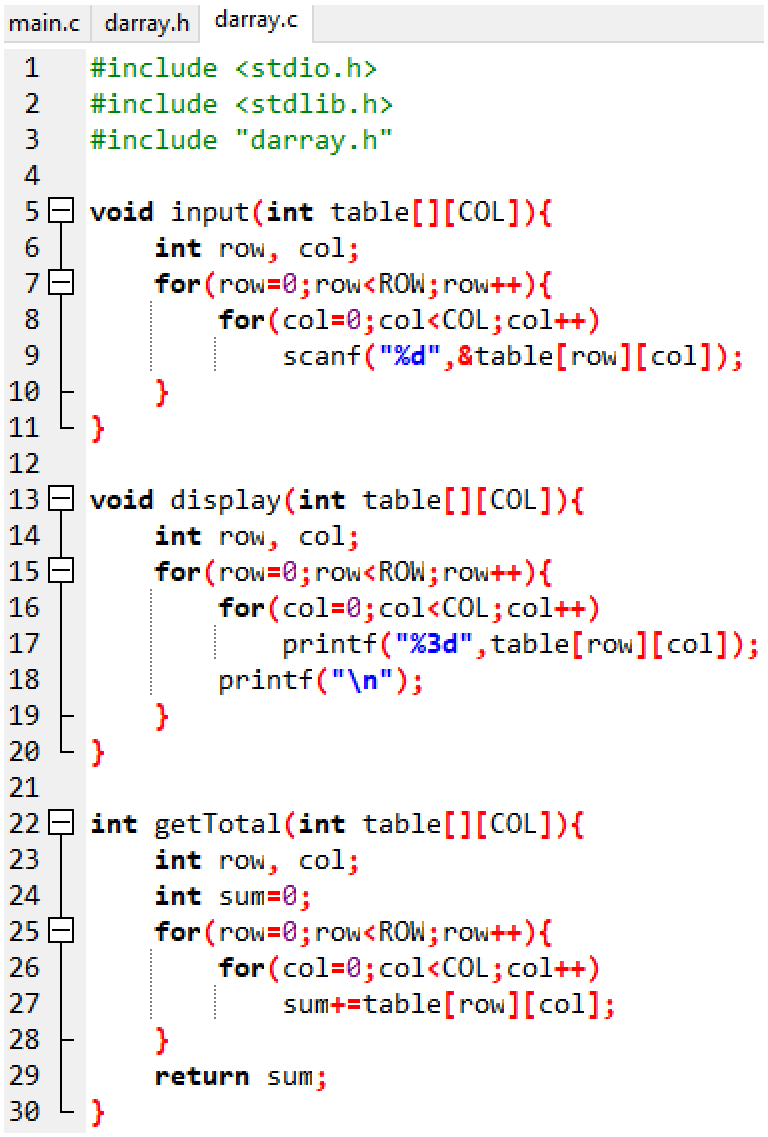
Passing 2D Array to a Function

When a 2D array is passed to functions, the size of the column is required. The reference to the first element is passed as an actual parameter.

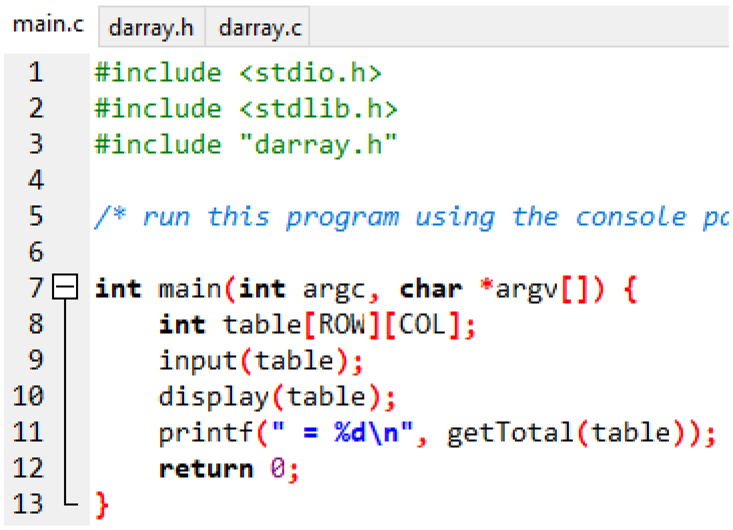
Here is an example:



```c
#define ROW 3
#define COL 4

void input(int table[][COL]);
void display(int table[][COL]);
int getTotal(int table[][COL]);
```

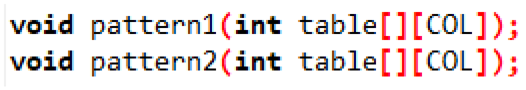


```c
#include <stdio.h>
#include <stdlib.h>
#include "darray.h"

void input(int table[][COL]){
    int row, col;
    for(row=0;row<ROW;row++){
        for(col=0;col<COL;col++)
            scanf("%d",&table[row][col]);
    }
}

void display(int table[][COL]){
    int row, col;
    for(row=0;row<ROW;row++){
        for(col=0;col<COL;col++)
            printf("%3d",table[row][col]);
        printf("\n");
    }
}

int getTotal(int table[][COL]){
    int row, col;
    int sum=0;
    for(row=0;row<ROW;row++){
        for(col=0;col<COL;col++)
            sum+=table[row][col];
    }
    return sum;
}
```

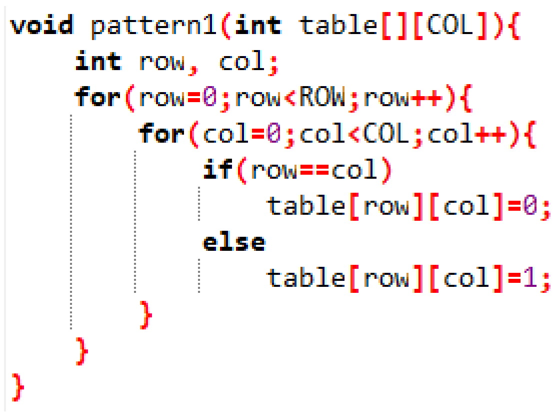


```
#include <stdio.h>
#include <stdlib.h>
#include "darray.h"

/* run this program using the console pc

int main(int argc, char *argv[]) {
    int table[ROW][COL];
    input(table);
    display(table);
    printf(" = %d\n", getTotal(table));
    return 0;
}
```

Here are more examples:

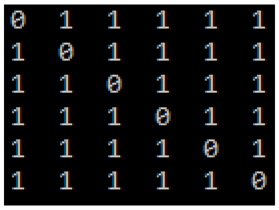


```
void pattern1(int table[][COL]);
void pattern2(int table[][COL]);
```



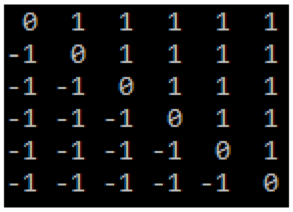
```
void pattern1(int table[][COL]){
    int row, col;
    for(row=0;row<ROW;row++){
        for(col=0;col<COL;col++){
            if(row==col)
                table[row][col]=0;
            else
                table[row][col]=1;
        }
    }
}
```

Output:



```
0  1  1  1  1  1
1  0  1  1  1  1
1  1  0  1  1  1
1  1  1  0  1  1
1  1  1  1  0  1
1  1  1  1  1  0
```

```
void pattern2(int table[][COL]){
    int row, col;
    for(row=0;row<ROW;row++){
        for(col=0;col<COL;col++){
            if(row==col)
                table[row][col]=0;
            else if(row<col)
                table[row][col]=1;
            else
                table[row][col]=-1;
        }
    }
}
```

Output: 

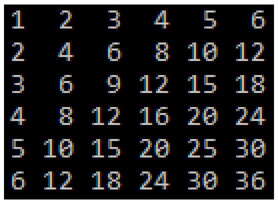
```
 0  1  1  1  1  1
-1  0  1  1  1  1
-1 -1  0  1  1  1
-1 -1 -1  0  1  1
-1 -1 -1 -1  0  1
-1 -1 -1 -1 -1  0
```

Practice Exercise (Ungraded)

Define the following function:

```
void multiplication(int table[][COL]);
```

Function multiplication() would generate the elements of the following 2D array:



```
1  2  3  4  5  6
2  4  6  8 10 12
3  6  9 12 15 18
4  8 12 16 20 24
5 10 15 20 25 30
6 12 18 24 30 36
```