Data Structures

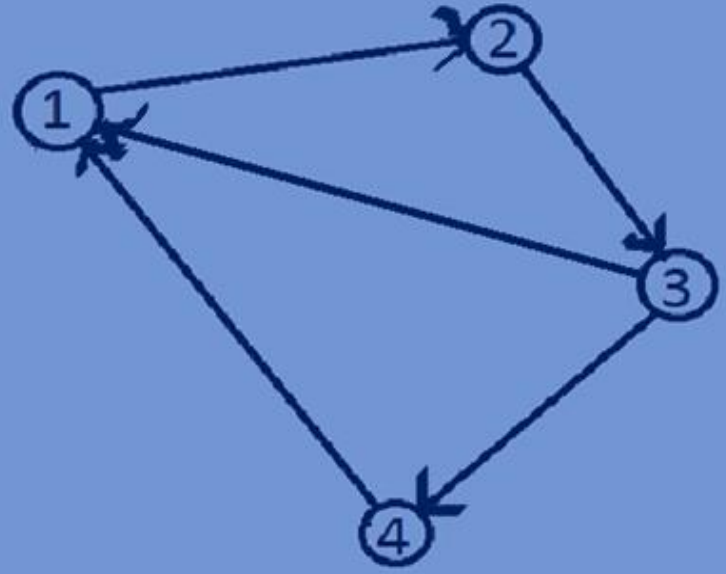
Implementation of a directed graph

Adjacency Matrix

Representation of an directed graph using adjacency matrix:

- In the adjacency matrix representation of a graph ,we represent a graph using Matrix of size V×V where V is the total no of vertices in a graph.So here we represent a graph using a two dimensional array of size V×V.Let the 2d array be array[][] and array[i][j]=1 indicates that there is a connection(edge) from vertex i to vertex j,since it is a directed graph we cant say whether there exist connection form vertex j to i until we know the value of array[j][i].

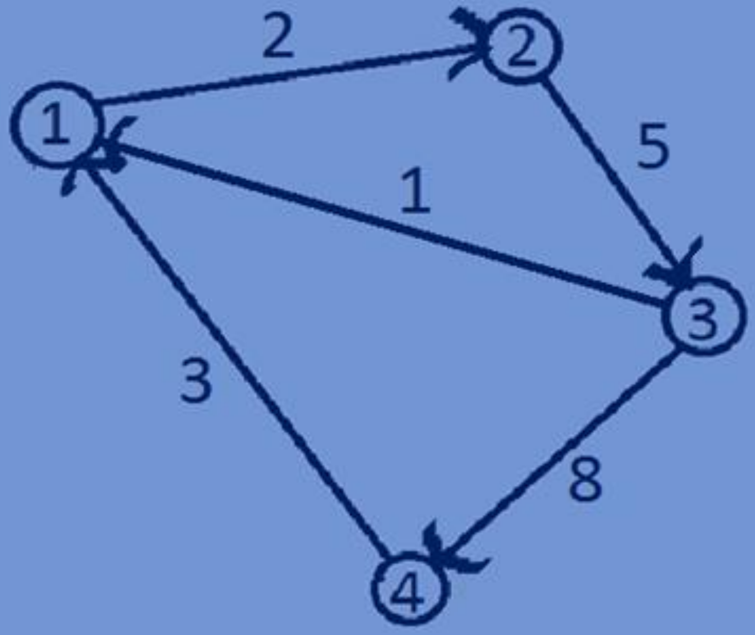
	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	1	0	0	1
4	1	0	0	0



Representation of a weighted directed graph using adjacency matrix:

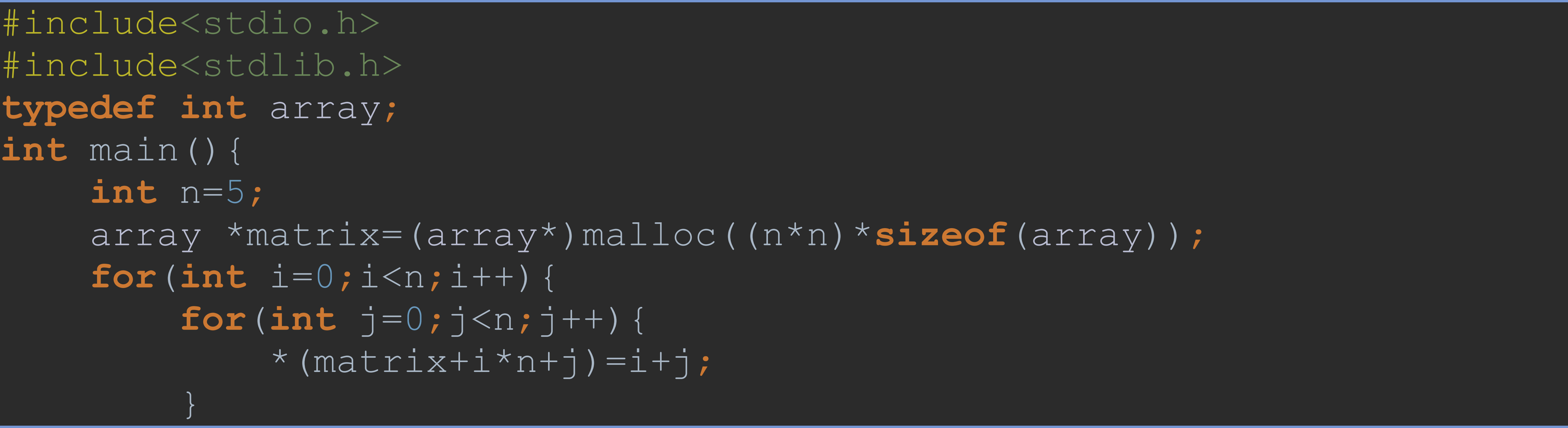
- We can also represent weighted directed graphs using adjacency matrix.
- If there is a weighted edge of weight ‘w’ from vertex ‘i’ to vertex ‘j’,then array[i][j]=’w’

	1	2	3	4
1	0	2	0	0
2	0	0	5	0
3	1	0	0	8
4	3	0	0	0

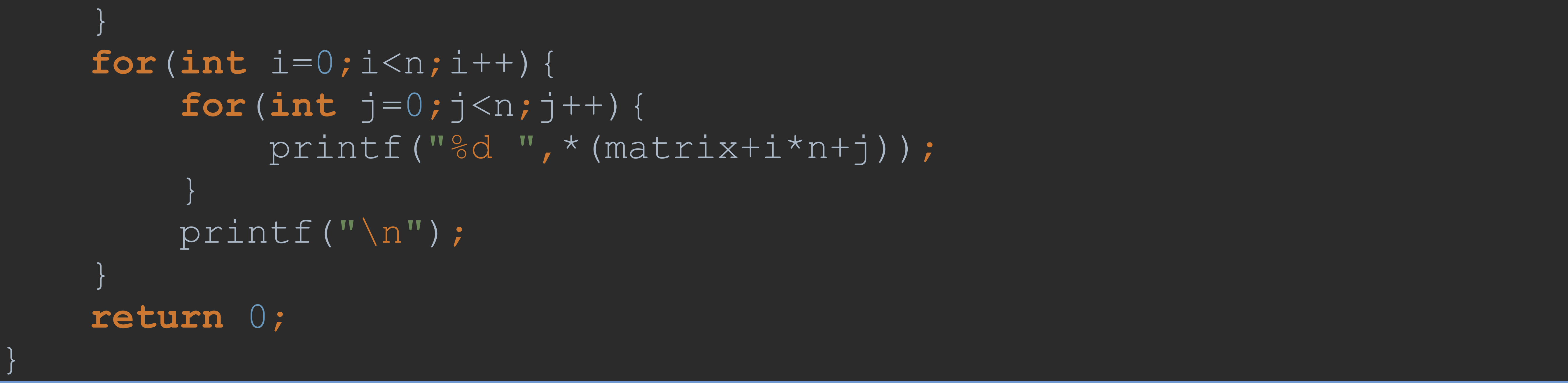


Allocating memory dynamically for a 2d array using single pointer:

- If there are ‘n’ vertices allocate n*n memory blocks because here we have to Construct a adjacency matrix of size n×n.
- In the following example, we are going to represent matrix of size 4×4 using A integer pointer.



```
#include<stdio.h>
#include<stdlib.h>
typedef int array;
int main(){
    int n=5;
    array *matrix=(array*)malloc((n*n)*sizeof(array));
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            *(matrix+i*n+j)=i+j;
        }
```



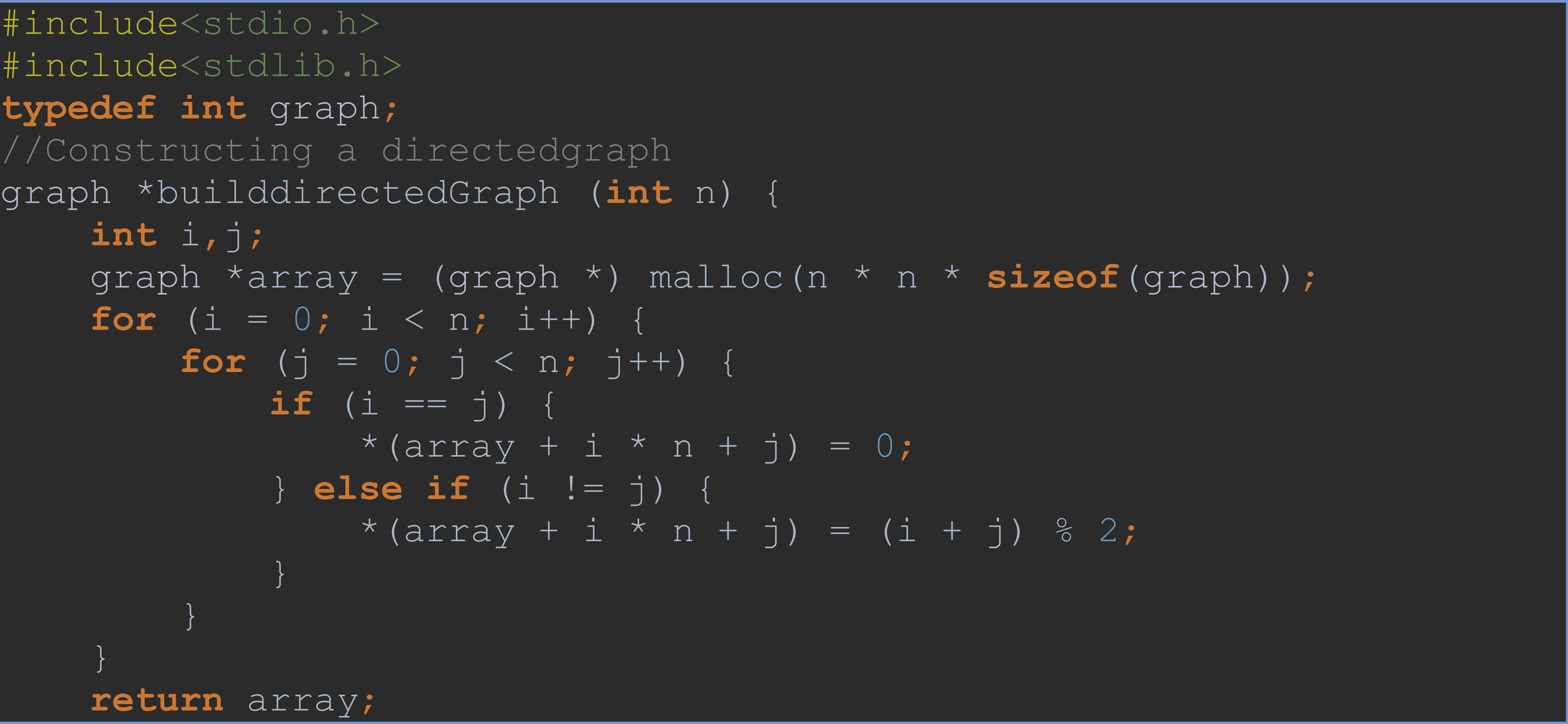
```
    }
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            printf("%d ",*(matrix+i*n+j));
        }
        printf("\n");
    }
    return 0;
}
```

Output:

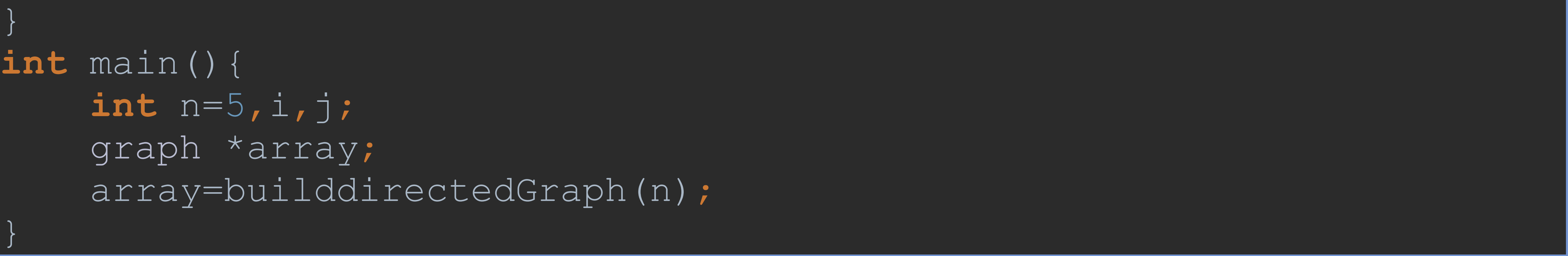
```
0  1  2  3  4

1  2  3  4  5

2  3  4  5  6

3  4  5  6  7

4  5  6  7  8
```

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

Function for implementing directed graph using Adjacency Matrix:



```c
#include<stdio.h>
#include<stdlib.h>
typedef int graph;
//Constructing a directedgraph
graph *builddirectedGraph (int n) {
    int i,j;
    graph *array = (graph *) malloc(n * n * sizeof(graph));
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i == j) {
                *(array + i * n + j) = 0;
            } else if (i != j) {
                *(array + i * n + j) = (i + j) % 2;
            }
        }
    }
    return array;
```



```
}
int main(){
     int n=5,i,j;
     graph *array;
     array=builddirectedGraph(n);
}
```

	0	1	2	3	4
0	0	1	0	1	0
1	1	0	1	0	1
2	0	1	0	1	0
3	1	0	1	0	1
4	0	1	0	1	0