CSE 06131223 ♦ CSE 06131224

Structured Programming

Lecture 12

Array in C (2)

Prepared by

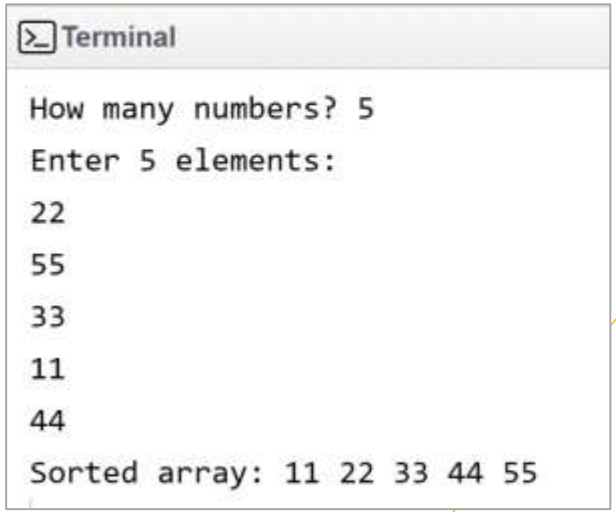
Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
www.mijanrahman.com

Contents

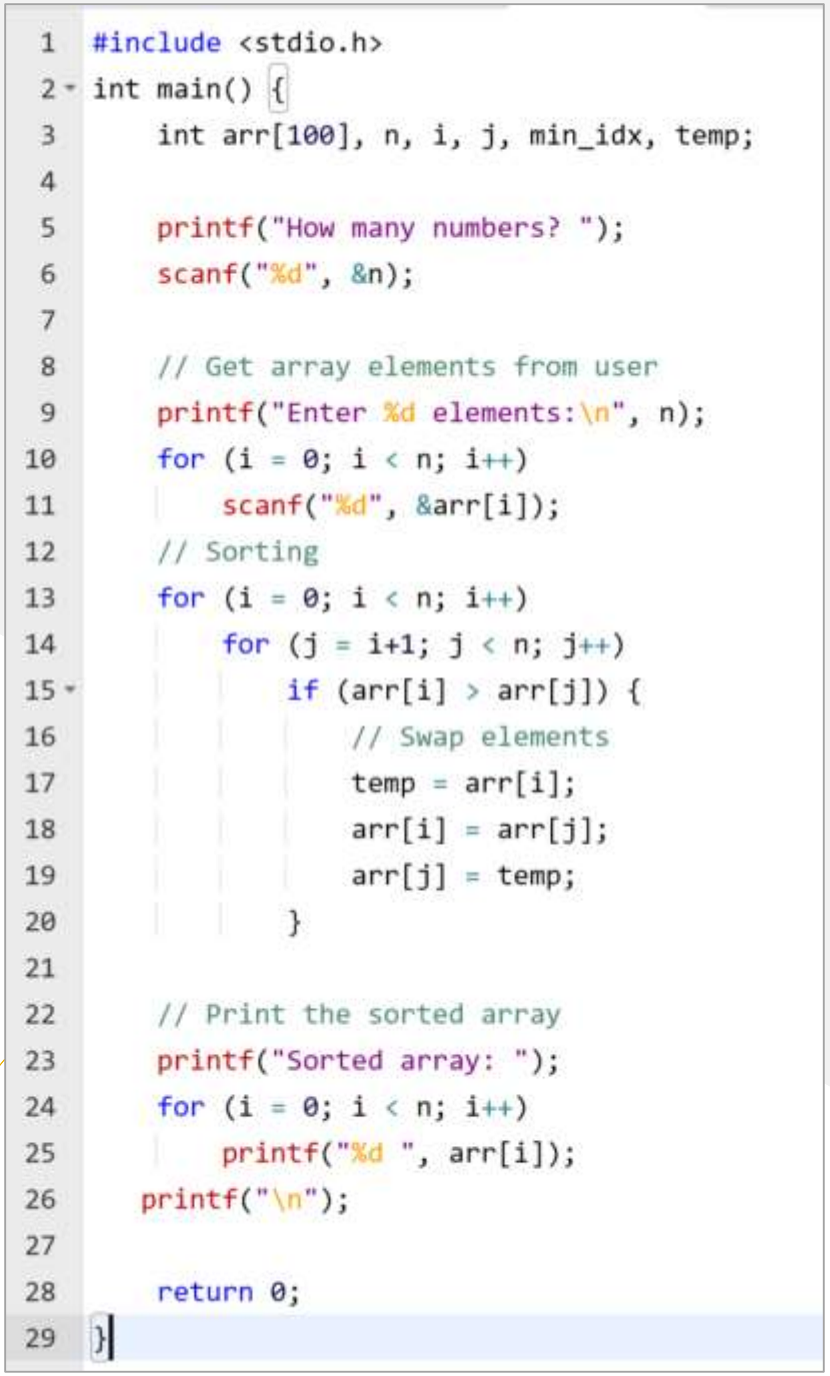
ARRAY IN C

C Program using array:

Sorting an array of N numbers.



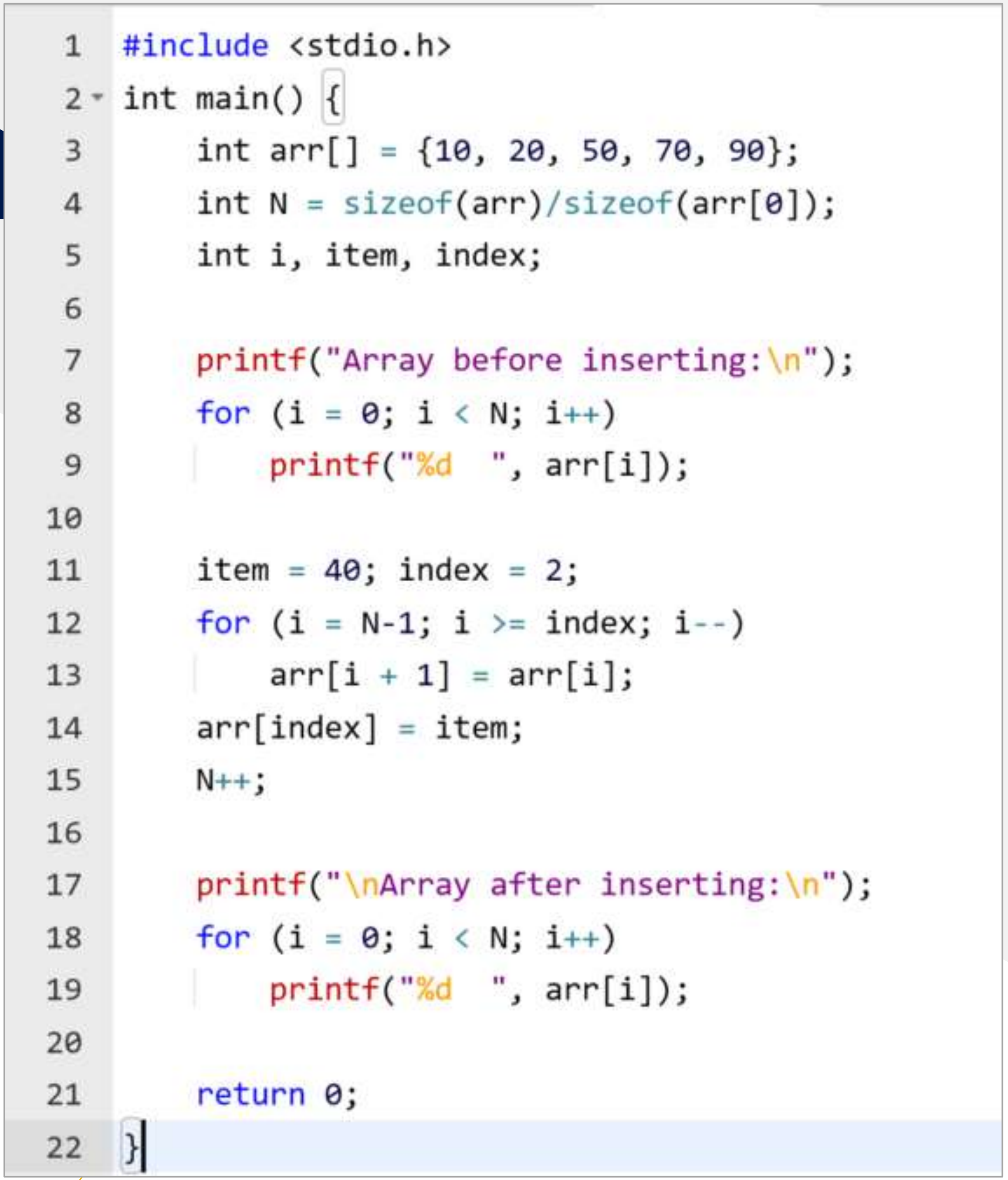
```
Terminal
How many numbers? 5
Enter 5 elements:
22
55
33
11
44
Sorted array: 11 22 33 44 55
```



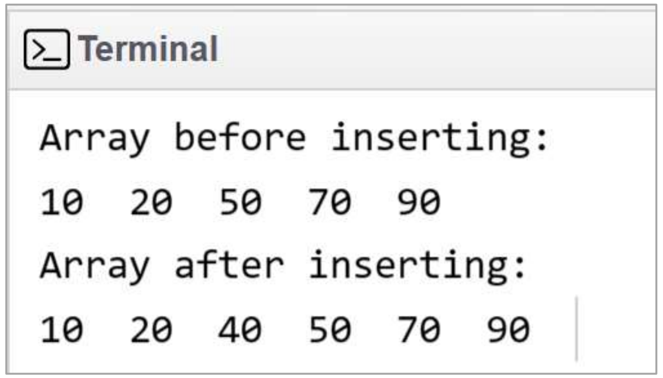
```c
#include <stdio.h>
int main() {
    int arr[100], n, i, j, min_idx, temp;

    printf("How many numbers? ");
    scanf("%d", &n);

    // Get array elements from user
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    // Sorting
    for (i = 0; i < n; i++)
        for (j = i+1; j < n; j++)
            if (arr[i] > arr[j]) {
                // Swap elements
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }

    // Print the sorted array
    printf("Sorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
   printf("\n");

    return 0;
}
```

C Program using array:

Inserting a new element into an array.



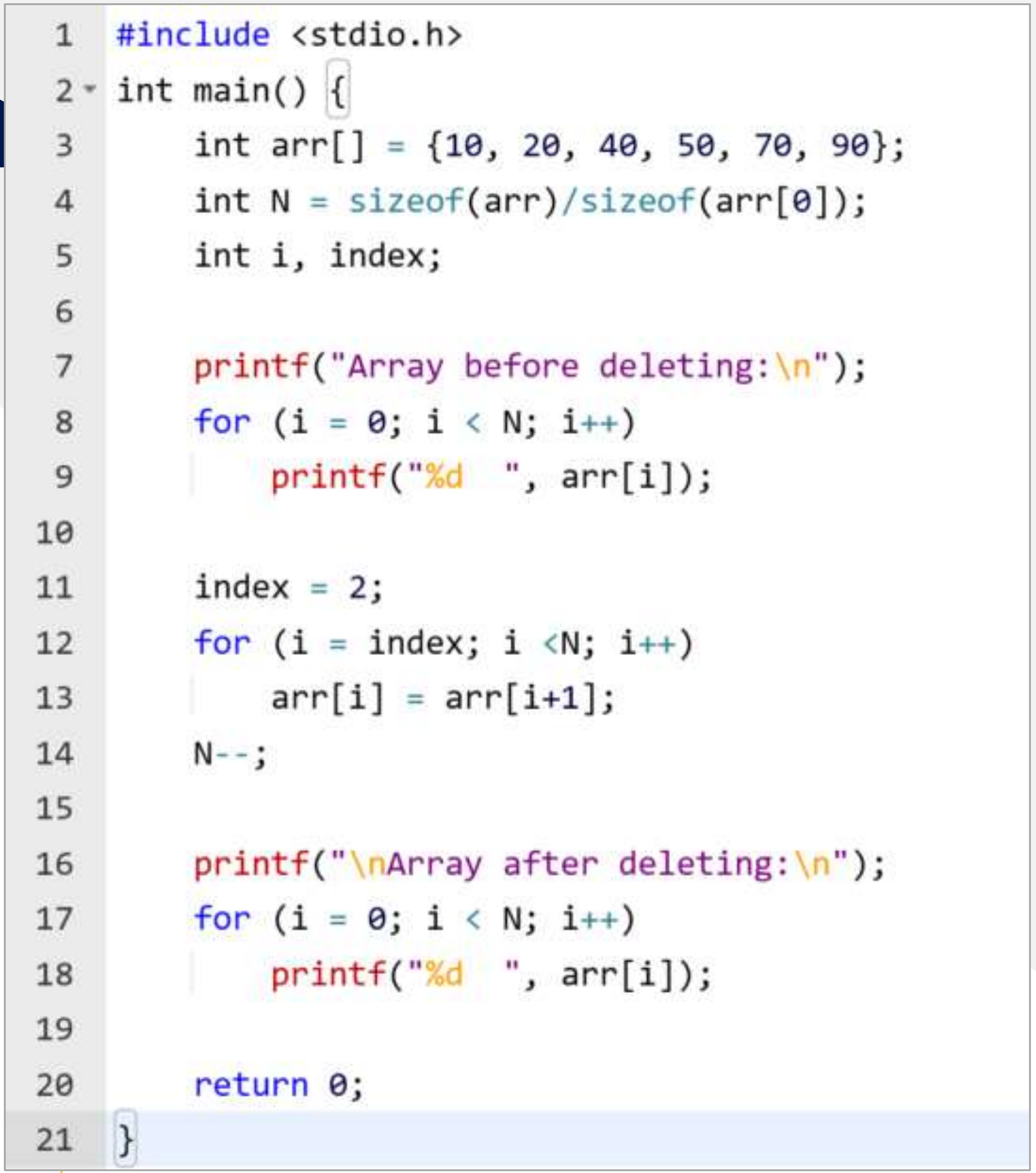
```c
#include <stdio.h>
int main() {
    int arr[] = {10, 20, 50, 70, 90};
    int N = sizeof(arr)/sizeof(arr[0]);
    int i, item, index;

    printf("Array before inserting:\n");
    for (i = 0; i < N; i++)
        printf("%d  ", arr[i]);

    item = 40; index = 2;
    for (i = N-1; i >= index; i--)
        arr[i + 1] = arr[i];
    arr[index] = item;
    N++;

    printf("\nArray after inserting:\n");
    for (i = 0; i < N; i++)
        printf("%d  ", arr[i]);

    return 0;
}
```



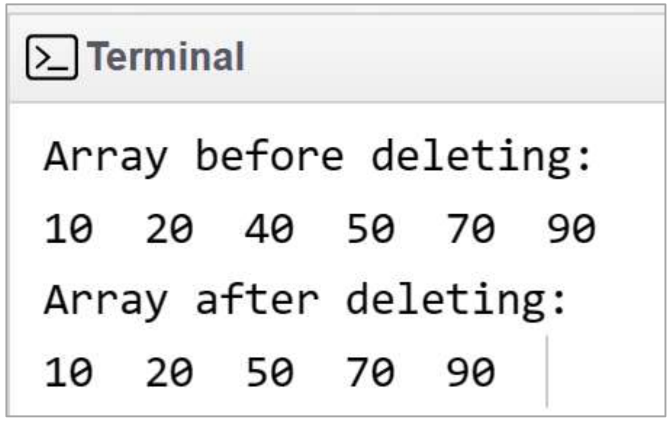
```
Array before inserting:
10  20  50  70  90
Array after inserting:
10  20  40  50  70  90
```

C Program using array:

Deleting an element from an array.



```c
#include <stdio.h>
int main() {
    int arr[] = {10, 20, 40, 50, 70, 90};
    int N = sizeof(arr)/sizeof(arr[0]);
    int i, index;

    printf("Array before deleting:\n");
    for (i = 0; i < N; i++)
        printf("%d  ", arr[i]);

    index = 2;
    for (i = index; i <N; i++)
        arr[i] = arr[i+1];
    N--;

    printf("\nArray after deleting:\n");
    for (i = 0; i < N; i++)
        printf("%d  ", arr[i]);

    return 0;
}
```



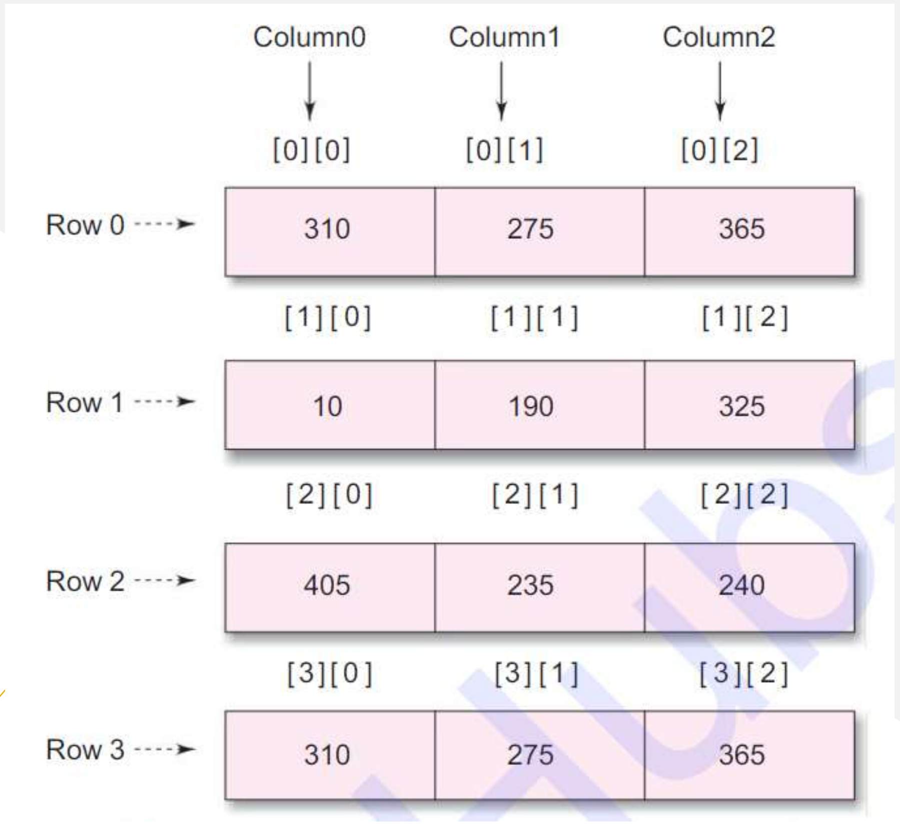
```
Array before deleting:
10  20  40  50  70  90
Array after deleting:
10  20  50  70  90
```

Two Dimensional (2D) Array in C

- The two-dimensional array can be defined as an array of arrays.
- **The 2D array is organized as matrices which can be represented as the collection of rows and columns.**
- However, 2D arrays are created to implement a relational database look alike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

Two Dimensional (2D) Array in C

- Two-dimensional arrays are stored in memory, as shown in the following figure.
- As with the single-dimensional arrays, each dimension of the array is indexed from zero to its maximum size minus one.
- **The first index selects the row and the second index selects the column within that row.**



Declaration of 2D Array in C

- **The syntax to declare the 2D array is given below:**

```
data-type arrayName[rows][columns];
```

- Consider the following example.

```
int arr[4][3];
```

Here, 4 is the number of rows, and 3 is the number of columns, and **arr** is the name of the array variable of integer type.

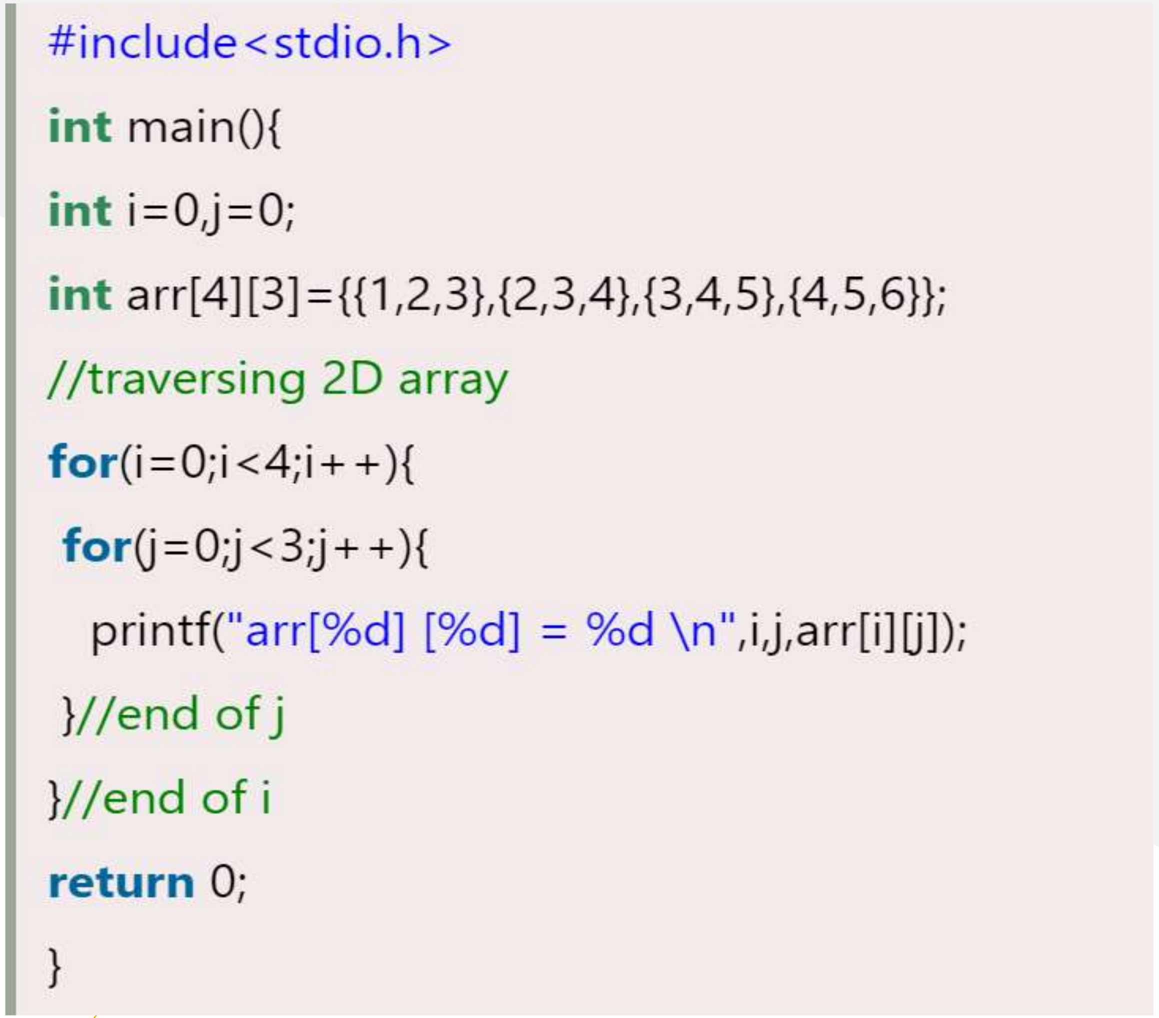
Initialization of 2D Array in C

- In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously.
- However, this will not work with 2D arrays.
- **We will have to define at least the second dimension of the array. The two-dimensional array can be declared and defined in the following way:**

```
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
int table[2][3] = {
                     {0,0,0},
                     {1,1,1}
                  };
```

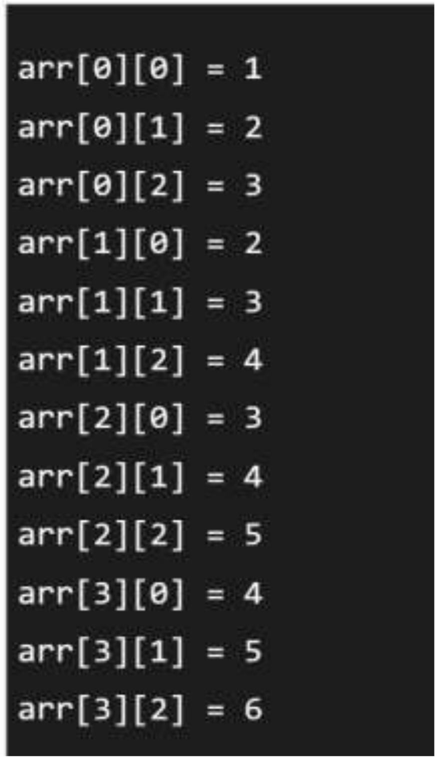
C Program using 2D array:

Traversing/accessing elements of 2D array.



```c
#include<stdio.h>
int main(){
int i=0,j=0;
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
//traversing 2D array
for(i=0;i<4;i++){
 for(j=0;j<3;j++){
   printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
 }//end of j
}//end of i
return 0;
}
```

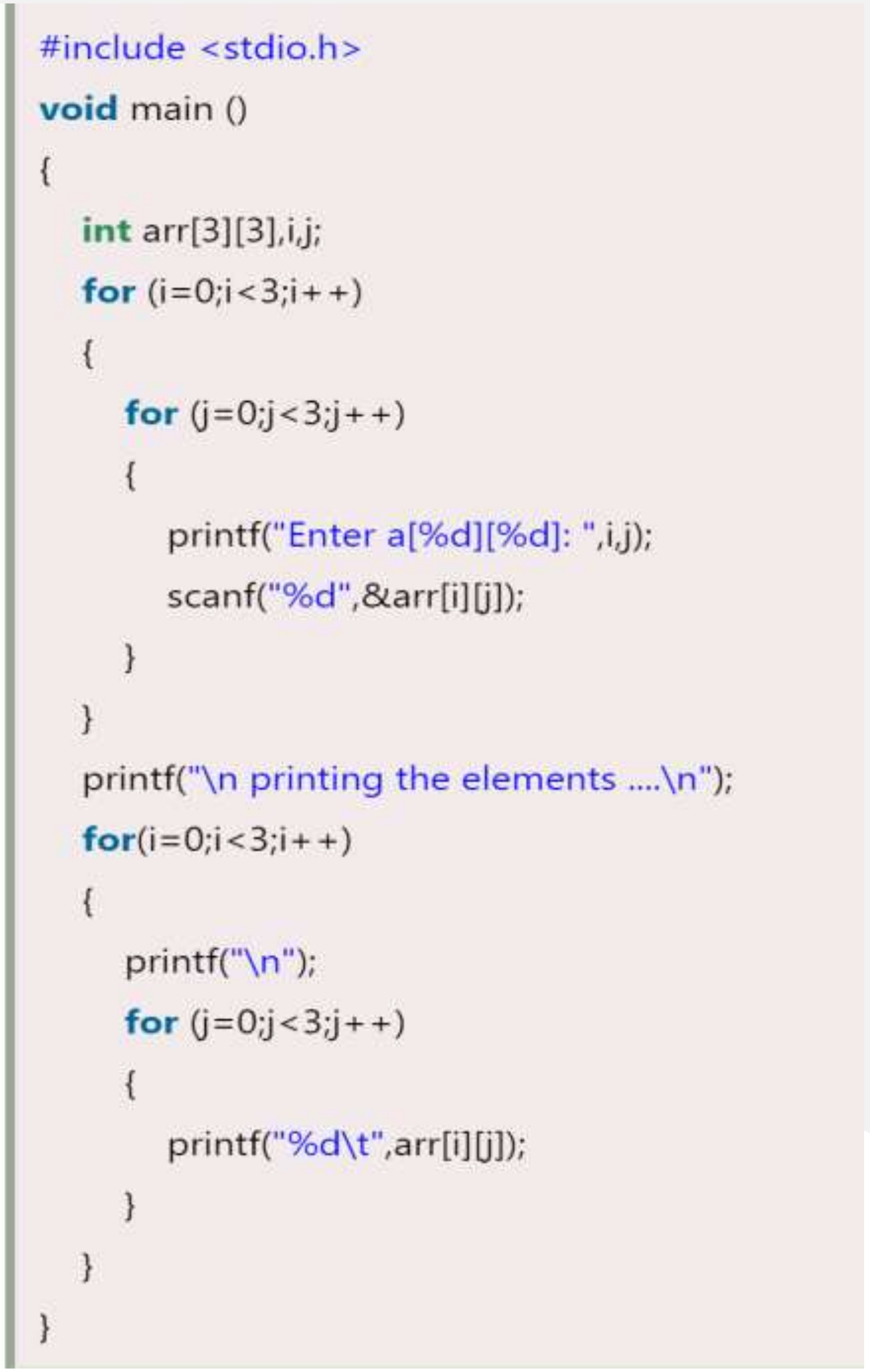
Output



```
arr[0][0] = 1
arr[0][1] = 2
arr[0][2] = 3
arr[1][0] = 2
arr[1][1] = 3
arr[1][2] = 4
arr[2][0] = 3
arr[2][1] = 4
arr[2][2] = 5
arr[3][0] = 4
arr[3][1] = 5
arr[3][2] = 6
```

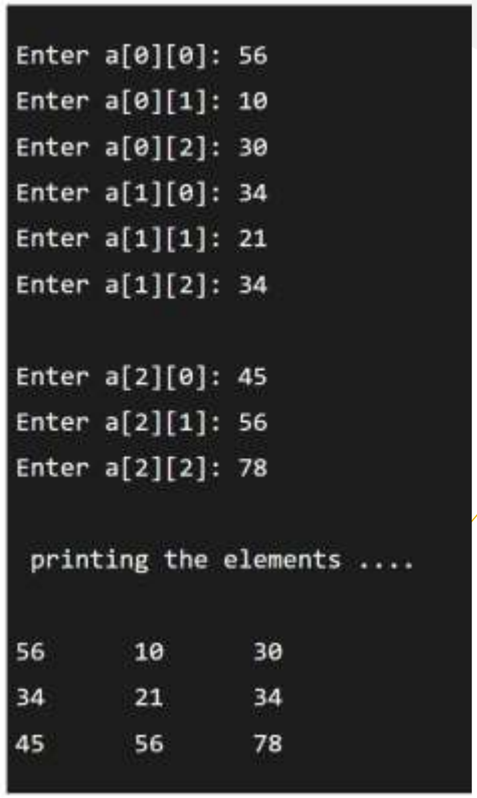
C Program using 2D array:

Printing elements of a matrix, receiving from input.



```c
#include <stdio.h>
void main ()
{
  int arr[3][3],i,j;
  for (i=0;i<3;i++)
  {
    for (j=0;j<3;j++)
    {
      printf("Enter a[%d][%d]: ",i,j);
      scanf("%d",&arr[i][j]);
    }
  }
  printf("\n printing the elements ....\n");
  for(i=0;i<3;i++)
  {
    printf("\n");
    for (j=0;j<3;j++)
    {
      printf("%d\t",arr[i][j]);
    }
  }
}
```

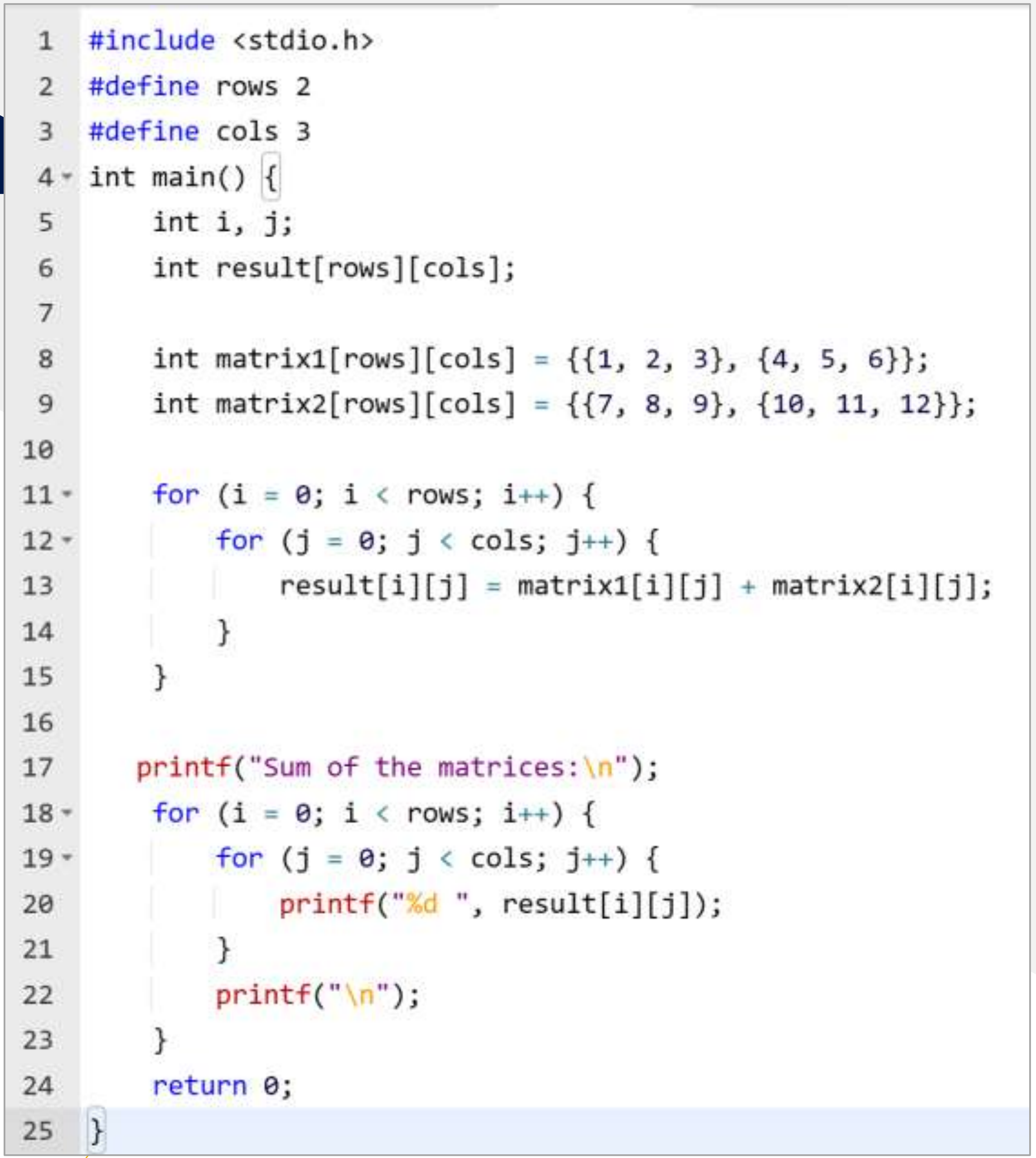
Output



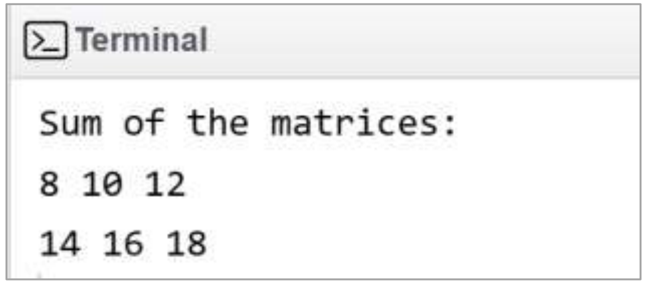
```
Enter a[0][0]: 56
Enter a[0][1]: 10
Enter a[0][2]: 30
Enter a[1][0]: 34
Enter a[1][1]: 21
Enter a[1][2]: 34

Enter a[2][0]: 45
Enter a[2][1]: 56
Enter a[2][2]: 78

 printing the elements ....

56      10      30
34      21      34
45      56      78
```

C Program using 2D array:

Adding two given matrices.



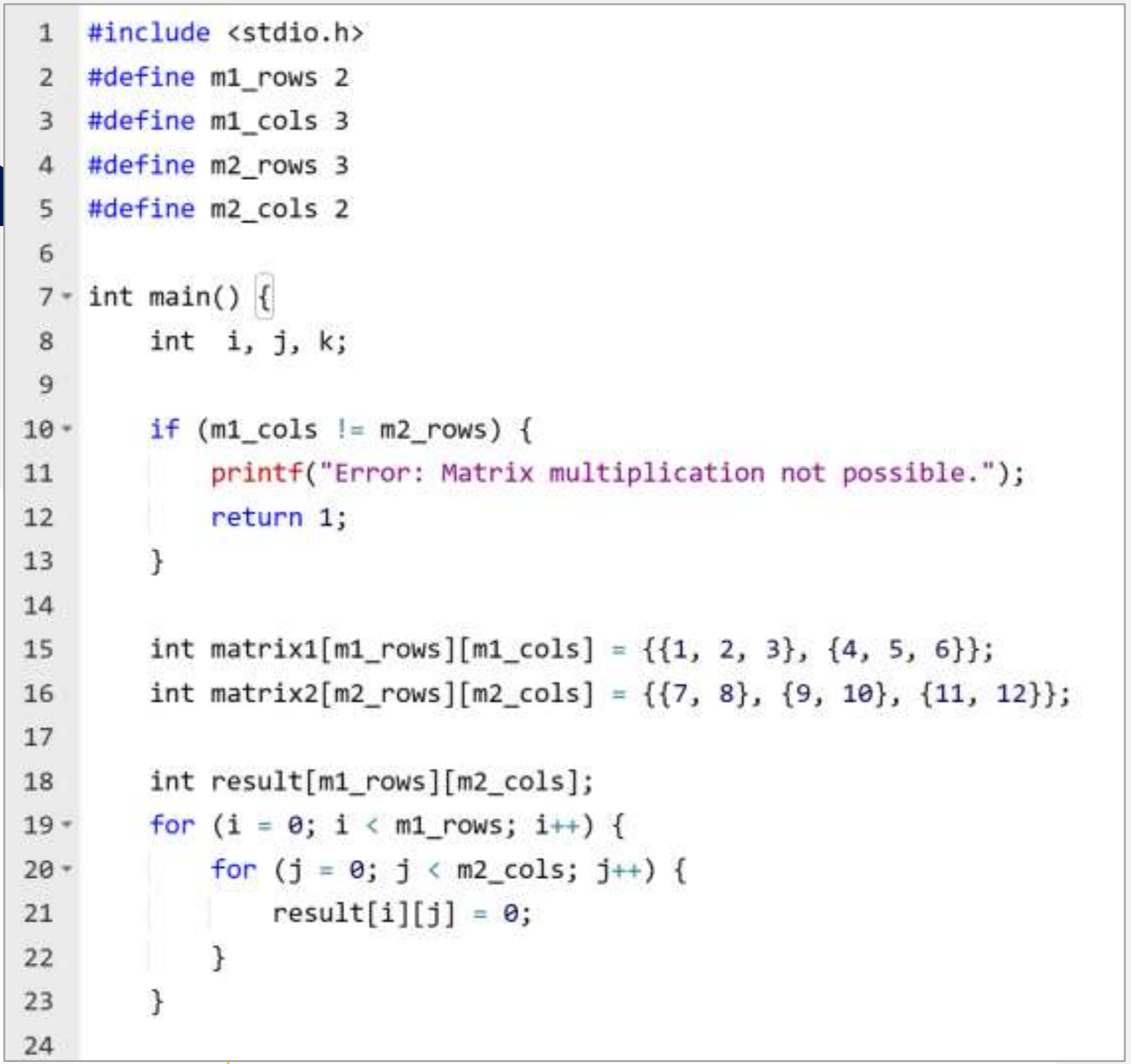
```c
#include <stdio.h>
#define rows 2
#define cols 3
int main() {
    int i, j;
    int result[rows][cols];

    int matrix1[rows][cols] = {{1, 2, 3}, {4, 5, 6}};
    int matrix2[rows][cols] = {{7, 8, 9}, {10, 11, 12}};

    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }

   printf("Sum of the matrices:\n");
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```



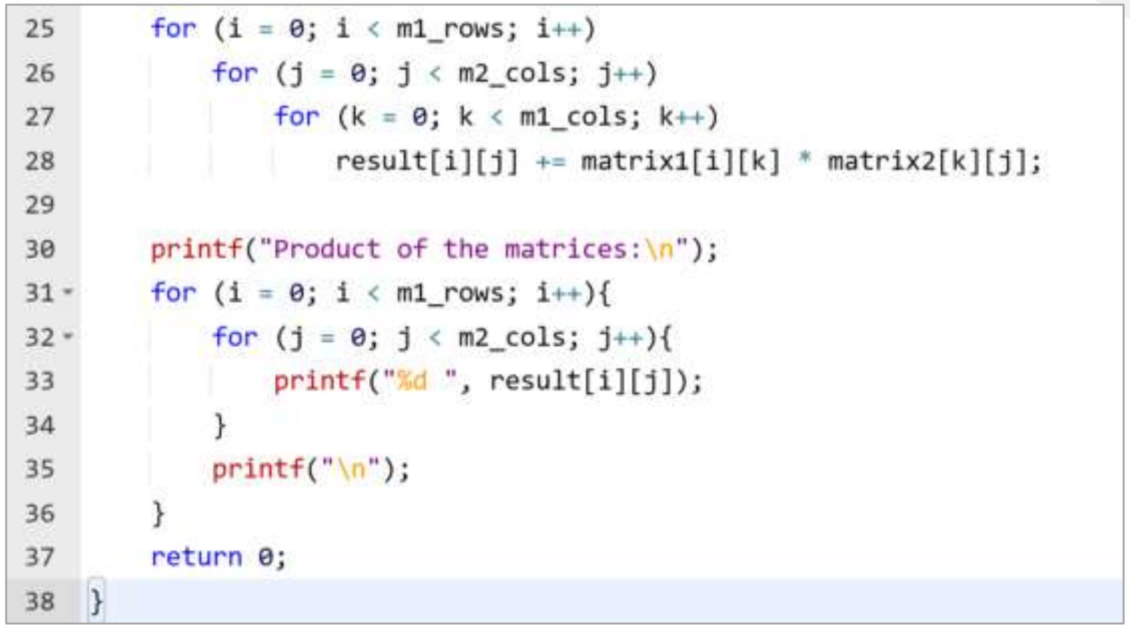
```
Sum of the matrices:
8 10 12
14 16 18
```

C Program using 2D array:

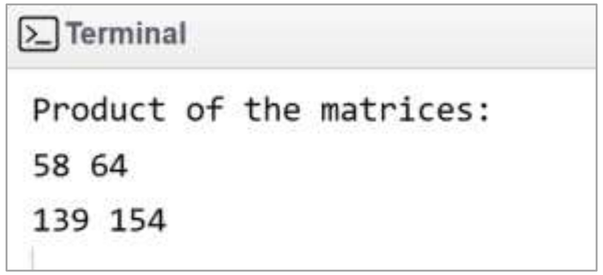
Multiplication of two given matrices.



```c
#include <stdio.h>
#define m1_rows 2
#define m1_cols 3
#define m2_rows 3
#define m2_cols 2

int main() {
    int  i, j, k;

    if (m1_cols != m2_rows) {
        printf("Error: Matrix multiplication not possible.");
        return 1;
    }

    int matrix1[m1_rows][m1_cols] = {{1, 2, 3}, {4, 5, 6}};
    int matrix2[m2_rows][m2_cols] = {{7, 8}, {9, 10}, {11, 12}};

    int result[m1_rows][m2_cols];
    for (i = 0; i < m1_rows; i++) {
        for (j = 0; j < m2_cols; j++) {
            result[i][j] = 0;
        }
    }

```



```c
    for (i = 0; i < m1_rows; i++)
        for (j = 0; j < m2_cols; j++)
            for (k = 0; k < m1_cols; k++)
                result[i][j] += matrix1[i][k] * matrix2[k][j];

    printf("Product of the matrices:\n");
    for (i = 0; i < m1_rows; i++){
        for (j = 0; j < m2_cols; j++){
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```



Terminal

```
Product of the matrices:
58 64
139 154
```

Character Arrays and Strings

- For example, the following array (or string of characters):

```
char name [20];
```

- **Initialization of strings:**
  - Because strings of characters are ordinary arrays they fulfill all their same rules. For example, if we want to initialize a string of characters with predetermined values we can do it just like any other array:

```
char mystring[] = { 'H', 'e', 'l', 'l', 'o', '\0’ };
```

  - Therefore we could initialize the string **mystring** with values by either of these two ways:

```
char mystring [] = { 'H', 'e', 'l', 'l', 'o', '\0' };
char mystring [] = "Hello";
```

Character Arrays and Strings

- **Assigning values to strings:**
- Since the ***lvalue*** of an assignation can only be an element of an array and not the entire array, it would be valid to assign a string of characters to an array of **char** using a method like this:

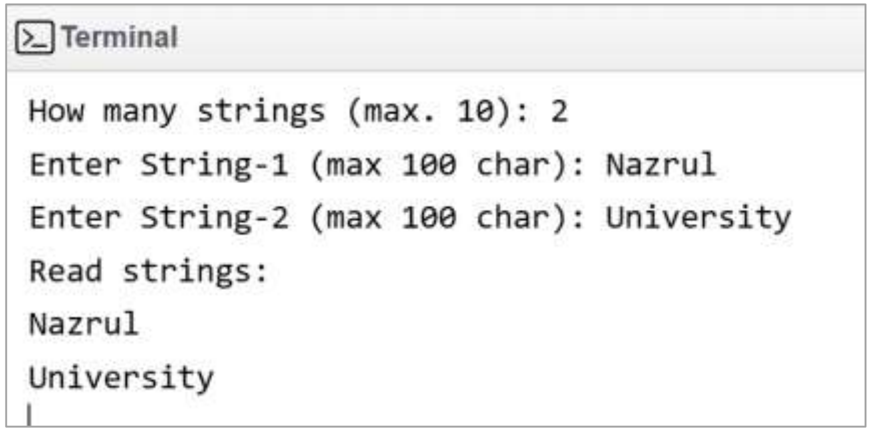
```
char mystring[10];
mystring[0] = 'H';
mystring[1] = 'e';
mystring[2] = 'l';
mystring[3] = 'l';
mystring[4] = 'o';
mystring[5] = '\0';
```

Character Arrays and Strings

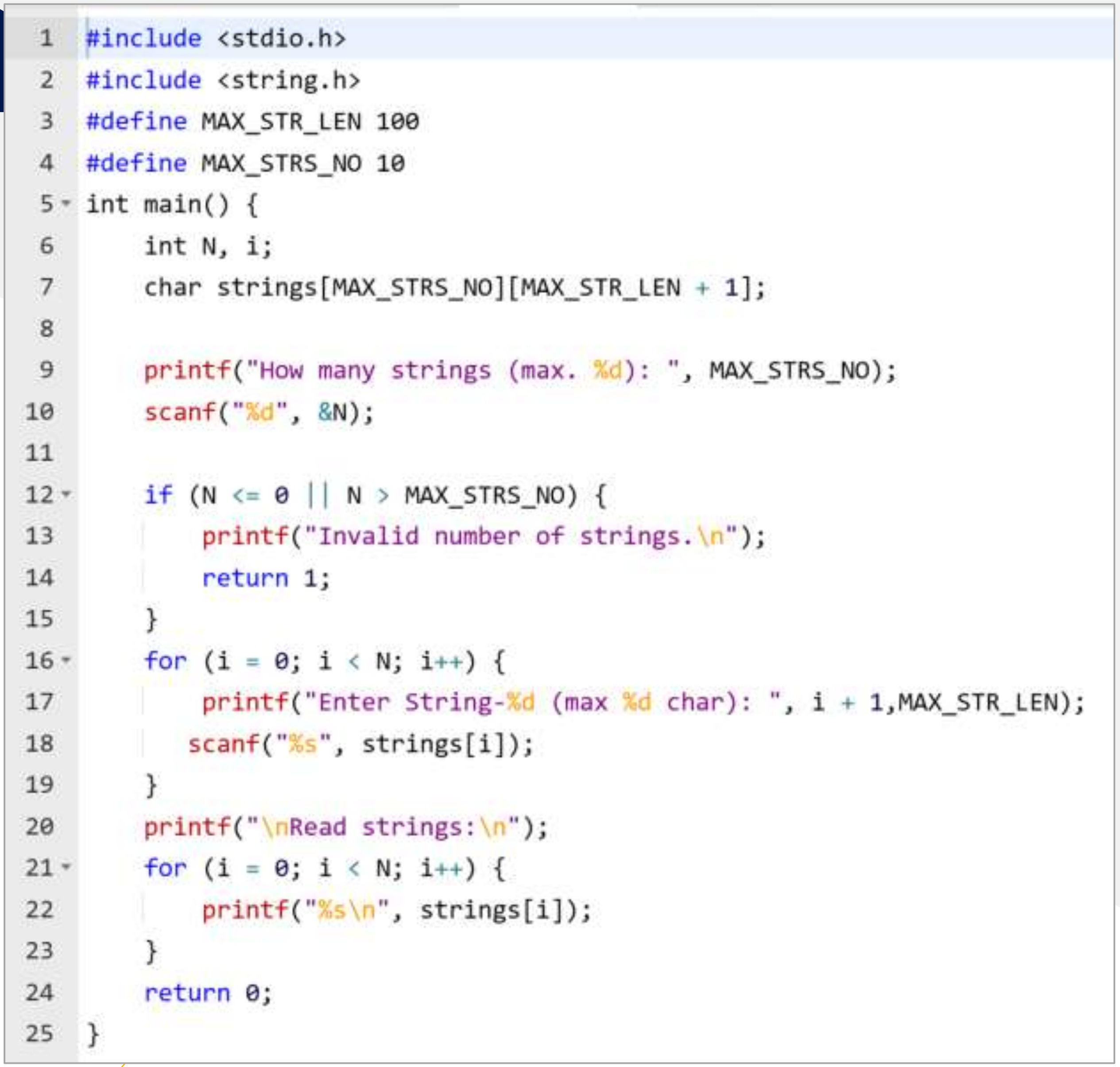
- **Reading and printing strings:**
- The input function scanf() can be used with %s format to read in a string of characters: For example:
- char address[10];
- scanf(“%s”, address);
- The output function printf() can be used with %s format to print the string to the terminal. For example:
- printf(“The given address is %s”, address);

C Program using Character Arrays and Strings:

Reading and printing of multiple strings.



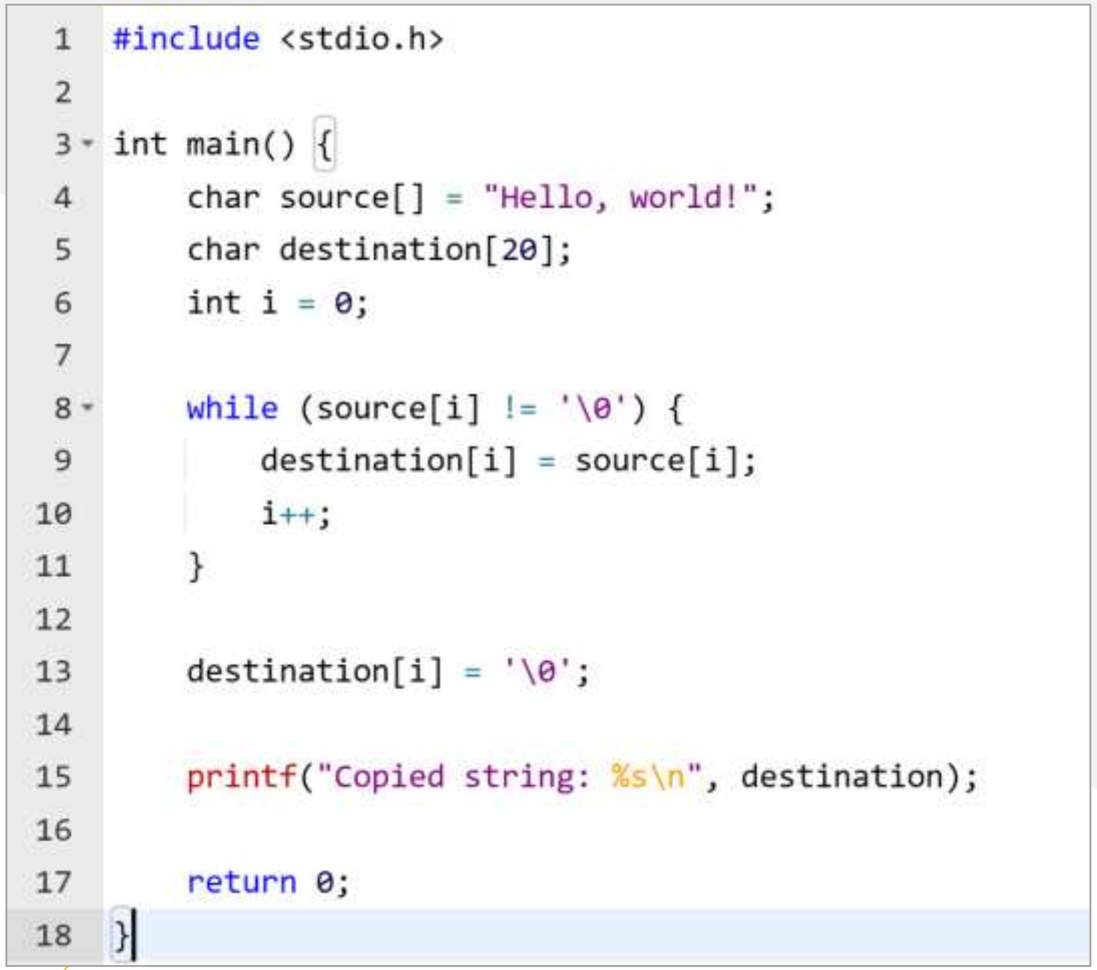
```
Terminal
How many strings (max. 10): 2
Enter String-1 (max 100 char): Nazrul
Enter String-2 (max 100 char): University
Read strings:
Nazrul
University
```



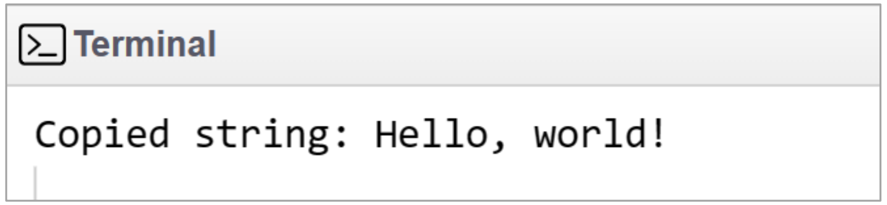
```c
#include <stdio.h>
#include <string.h>
#define MAX_STR_LEN 100
#define MAX_STRS_NO 10
int main() {
    int N, i;
    char strings[MAX_STRS_NO][MAX_STR_LEN + 1];

    printf("How many strings (max. %d): ", MAX_STRS_NO);
    scanf("%d", &N);

    if (N <= 0 || N > MAX_STRS_NO) {
        printf("Invalid number of strings.\n");
        return 1;
    }
    for (i = 0; i < N; i++) {
        printf("Enter String-%d (max %d char): ", i + 1,MAX_STR_LEN);
      scanf("%s", strings[i]);
    }
    printf("\nRead strings:\n");
    for (i = 0; i < N; i++) {
        printf("%s\n", strings[i]);
    }
    return 0;
}
```

C Program using Character Arrays and Strings:

Copying one string into another string.



```c
#include <stdio.h>

int main() {
    char source[] = "Hello, world!";
    char destination[20];
    int i = 0;

    while (source[i] != '\0') {
        destination[i] = source[i];
        i++;
    }

    destination[i] = '\0';

    printf("Copied string: %s\n", destination);

    return 0;
}
```

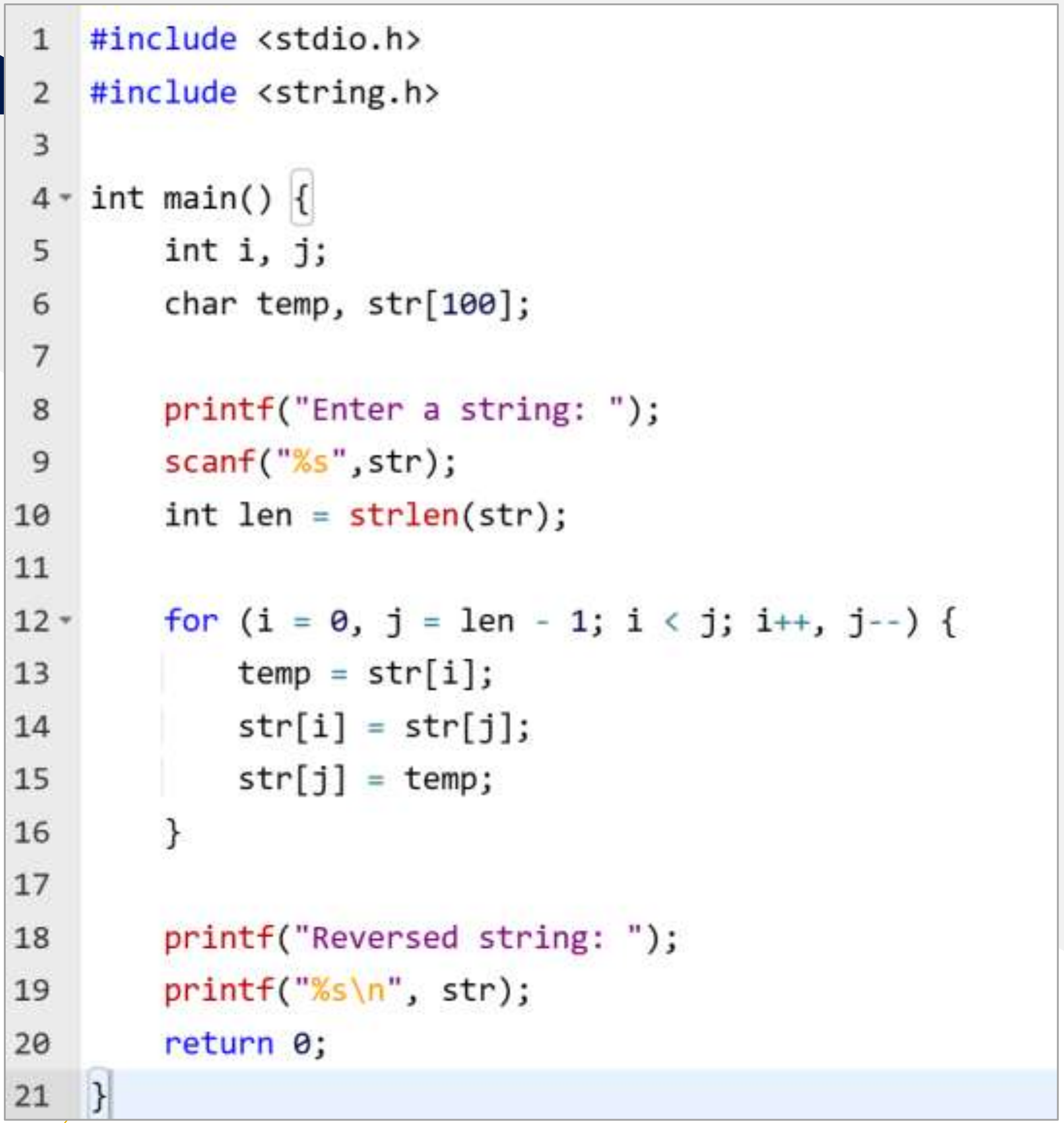


Terminal

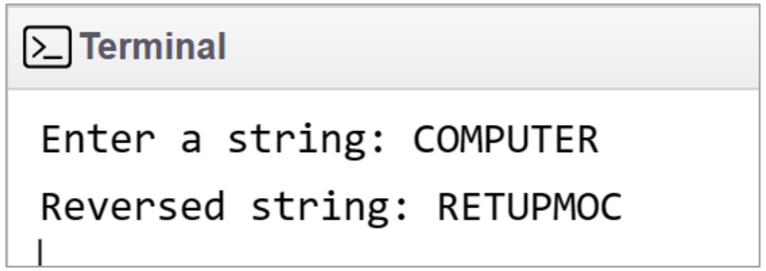
```
Copied string: Hello, world!
```

C Program using Character Arrays and Strings:

Reverse a string.



```c
#include <stdio.h>
#include <string.h>

int main() {
    int i, j;
    char temp, str[100];

    printf("Enter a string: ");
    scanf("%s",str);
    int len = strlen(str);

    for (i = 0, j = len - 1; i < j; i++, j--) {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }

    printf("Reversed string: ");
    printf("%s\n", str);
    return 0;
}
```



```
Enter a string: COMPUTER
Reversed string: RETUPMOC
```

? THE END