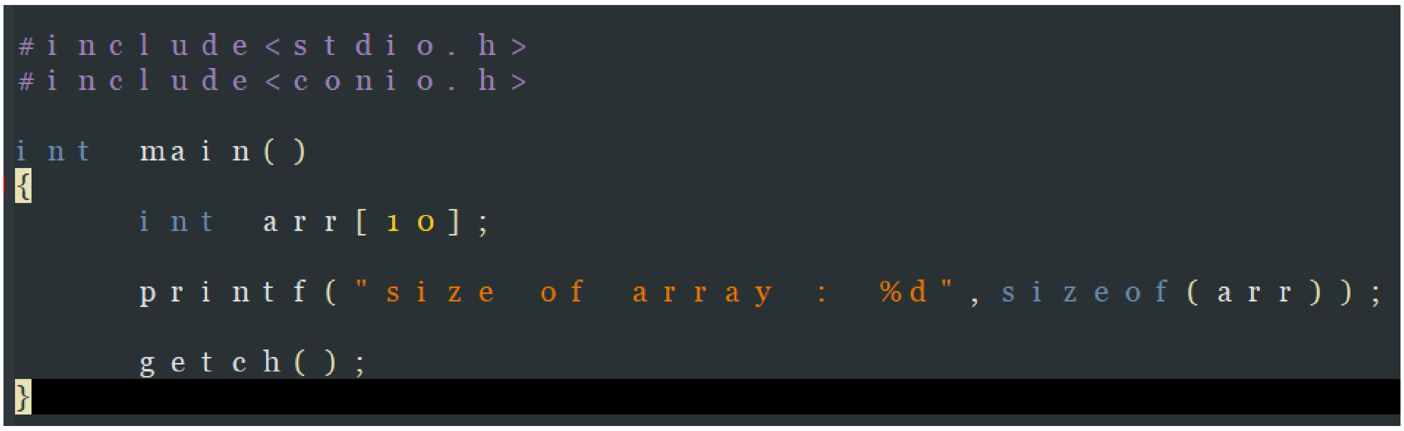
Array In C Programming:

In C programming language a data structure called array. An array can store a fixed size it means the sequential collection of elements of the same type. An array can be defined as collection of similar data items under single array variable name. All arrays consists of contiguous memory locations. The lowest address correspondences to the first element and highest address to the last element.

To declare an array in C programming language. A programmer specifies the type of elements and the number of elements required by an array as follows.



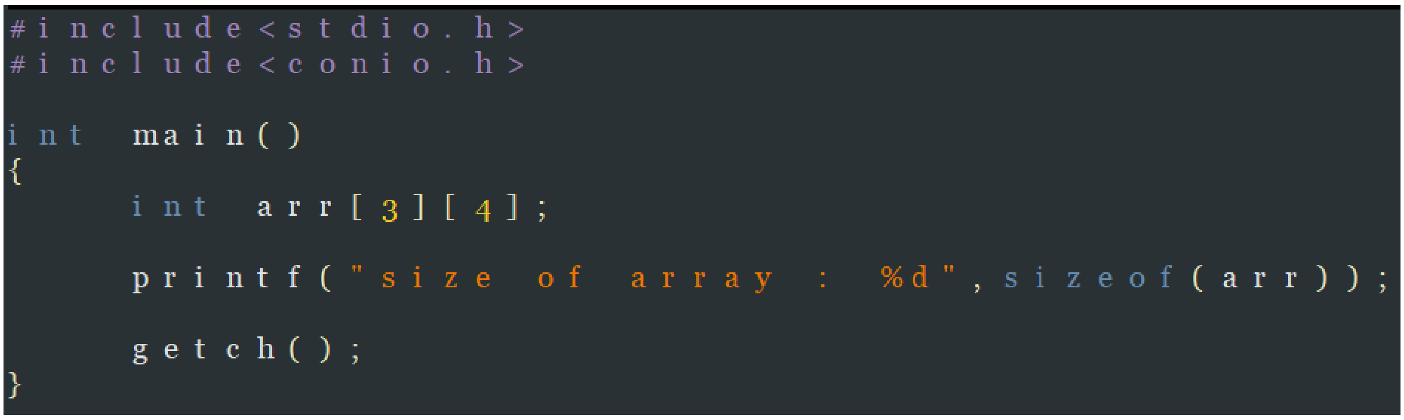
```
#include<stdio.h>
#include<conio.h>

int main()
{
	int arr[10];

	printf("size of array : %d",sizeof(arr));

	getch();
}
```

In the above program you can se we declare an array. “int arr[10];” in this the “int” is the data type of the array. so that we can only give integer type of data to that particular array. “arr” is the array variable name to define the identity of the array. “[10]” in this section we can give a required value. This value can define how many numbers of elements the particular array can hold. In the above example the array can hold 10 elements.

An array size specification is optional if the declaration and initialization is done at the same time. An element is accessed by indexing the array name. this is done by placing the index of the element within square brackets after the name of the array.

In simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is I essence a list of one-dimensional arrays. To declare a two dimensional integer array of size x, y would write something as follows.

Array is a linear collection of similar elements. Array is also known as subscript variable.



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

int  main()
{
     int  arr[3][4];

     printf("size of array :  %d",sizeof(arr));

     getch();
}
```

int arr[3][4]; “int” is the datatype, arr is the array variable name, [3][4] defines the row and column size of the array. In the above program the 3 is row and 4 is column size of the particular array.

Array Declaration Rule:

1. Suppose if you write like “int a[];” it is an error. because user must enter a value for the size of the array.

2. Suppose if you write like “int a[5];” it means the total number of variables in array. it is not an index value. User must enter only natural numbers.

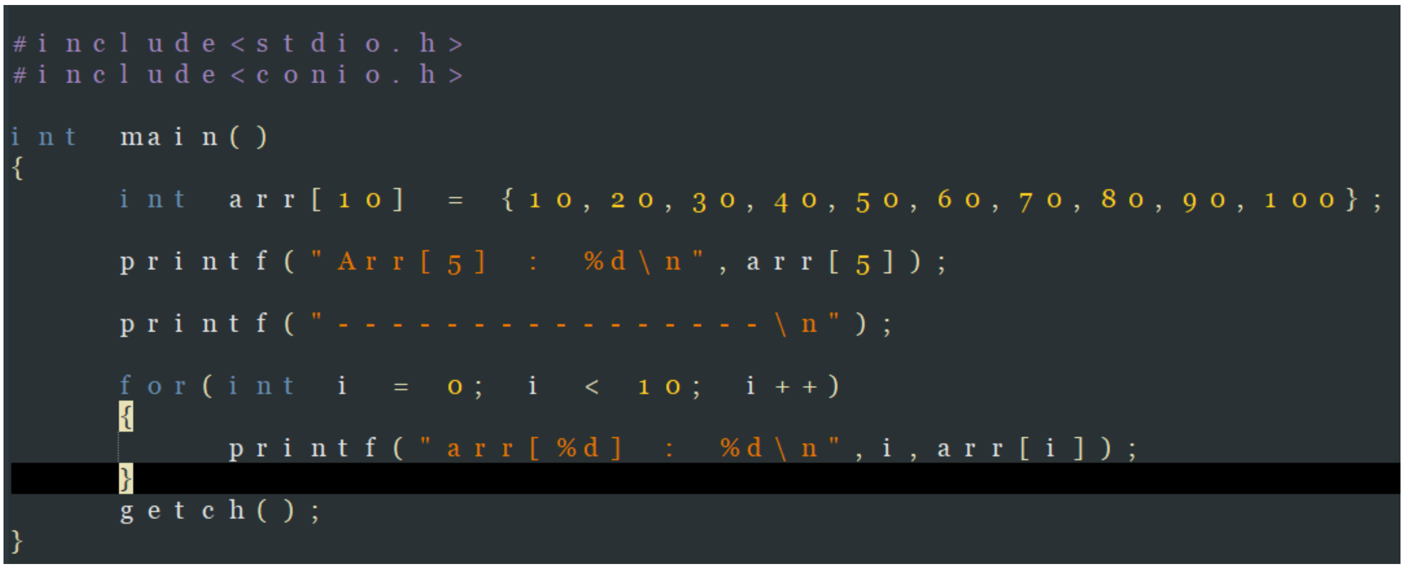
3. Suppose if you write like “int a[5];” and local array when not initialized any values it contains garbage values by default. Whatever is the size of the array it always consumes memory in a sequential; fashion.

4. You can initialize an array during declaration like this “int a[5] = {1,2,3,4,5};”.

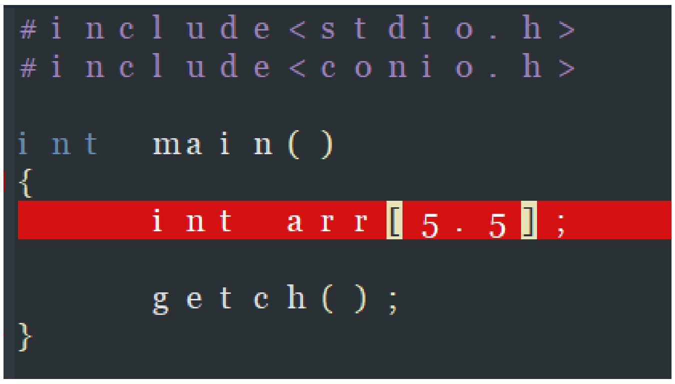
5. You cannot initialize an array during declaration more than its size. Example: int a[5] = {1,2,3,4,5,6,7,8,9,10};   //This Is A Wrong Statement.

6. You can initialize an array during declaration with lesser than the size of an array. Example: int a[5] = {1,2,3};   // the remaining space of the array will contain 0.

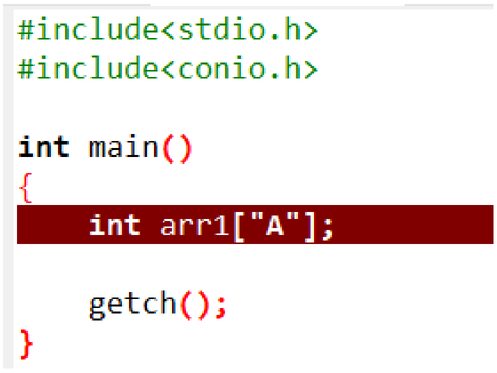
7. You cannot initialize the range of an array in floating type. Like “int arr[5.5];” “int arr[“A”];”



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

int  main()
{
     int  arr[10]  =  {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};

     printf("Arr[5]  :  %d\n", arr[5]);

     printf("------------------\n");

     for(int  i  =  0;  i  <  10;  i++)
     {
          printf("arr[%d]  :  %d\n", i, arr[i]);
     }
     getch();
}
```



```
#include<stdio.h>
#include<conio.h>

int  main()
{
	int  arr[5.5];

	getch();
}
```

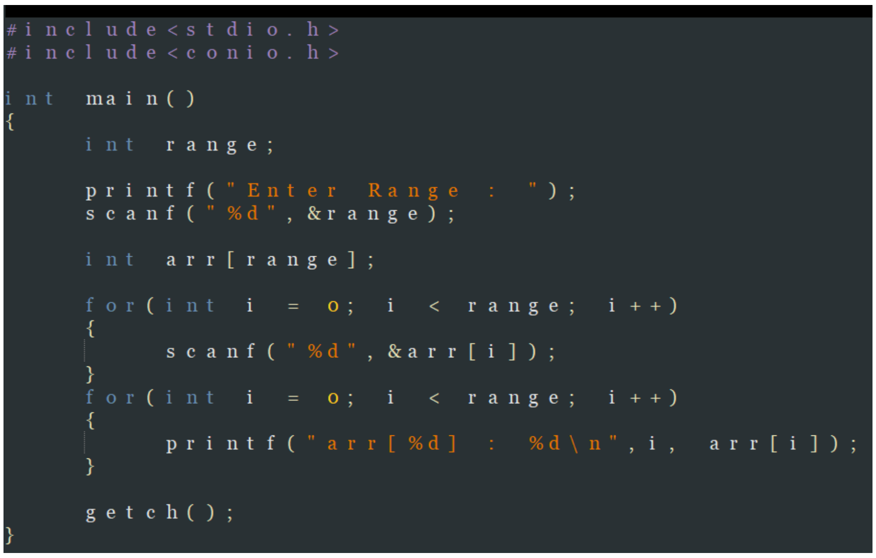


```
#include<stdio.h>
#include<conio.h>

int main()
{
	int arr1["A"];

	getch();
}
```

Dimensional In Array:

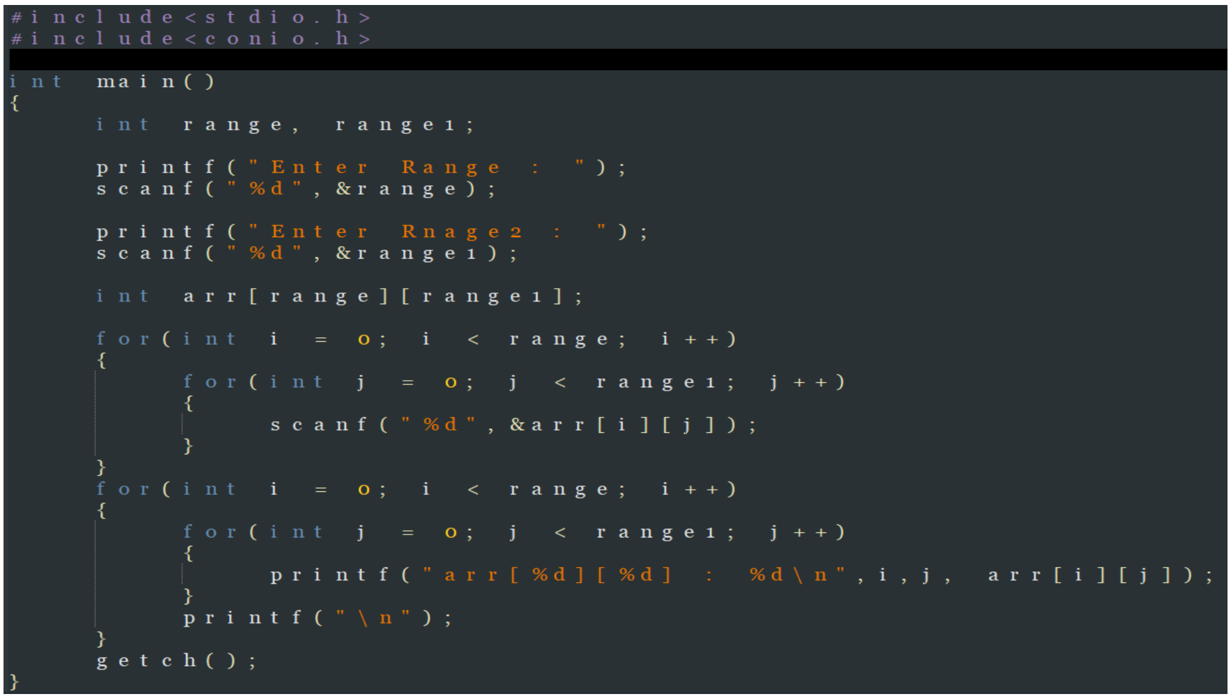
Dimension defines the direction of the array. You can create any dimension of array in mathematically. As a beginner programmer we use “One-Dimensional Array”, “Two-Dimensional Array”.

One-Dimensional Array:



```
#include<stdio.h>
#include<conio.h>

int  main()
{
	int  range;

	printf("Enter  Range  :  ");
	scanf("%d",&range);

	int  arr[range];

	for(int  i  =  0;  i  <  range;  i++)
	{
		scanf("%d",&arr[i]);
	}
	for(int  i  =  0;  i  <  range;  i++)
	{
		printf("arr[%d]  :  %d\n",i,  arr[i]);
	}

	getch();
}
```

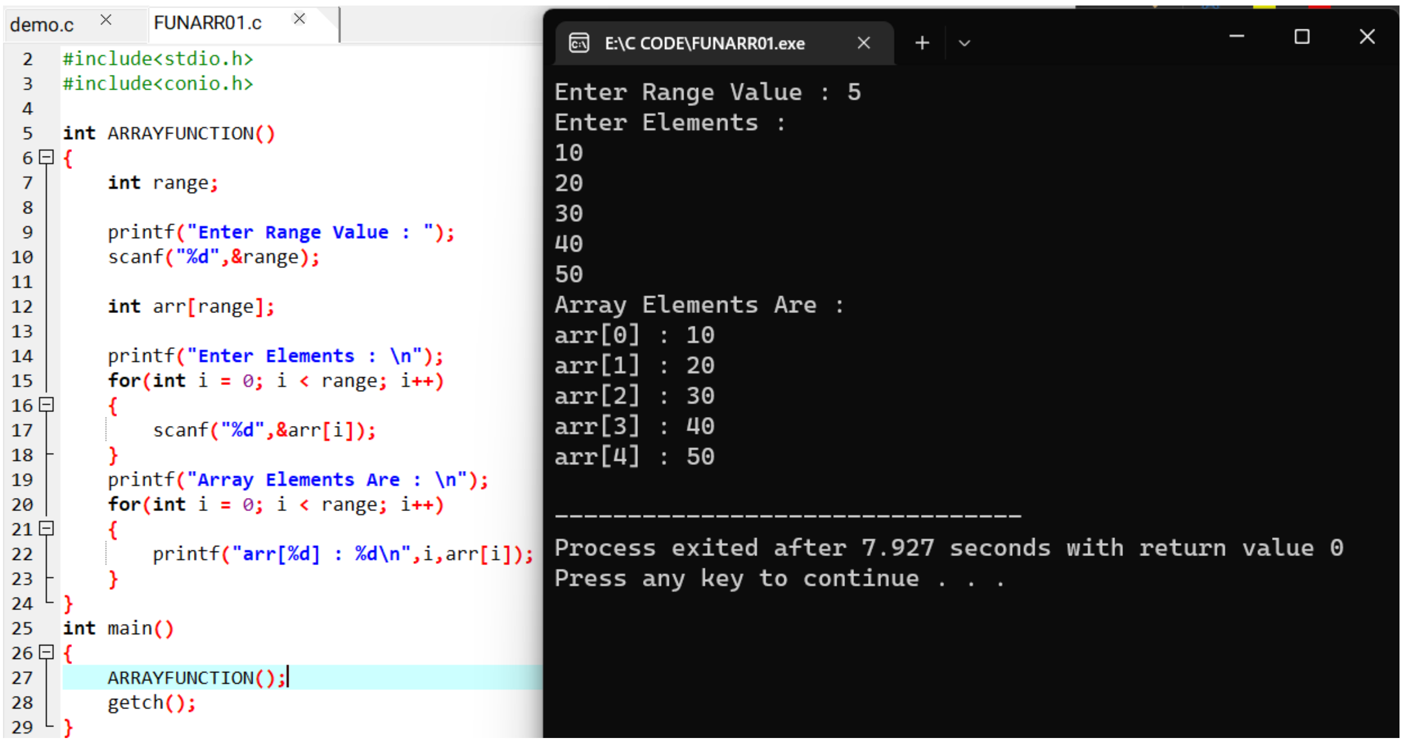
Two-Dimensional Array:



```
#include<stdio.h>
#include<conio.h>

int  main()
{
	int  range,  range1;

	printf("Enter  Range  :  ");
	scanf("%d",&range);

	printf("Enter  Rnage2  :  ");
	scanf("%d",&range1);

	int  arr[range][range1];

	for(int  i  =  0;  i  <  range;  i++)
	{
		for(int  j  =  0;  j  <  range1;  j++)
		{
			scanf("%d",&arr[i][j]);
		}
	}
	for(int  i  =  0;  i  <  range;  i++)
	{
		for(int  j  =  0;  j  <  range1;  j++)
		{
			printf("arr[%d][%d]  :  %d\n",i,j,  arr[i][j]);
		}
		printf("\n");
	}
	getch();
}
```

Function & Array:

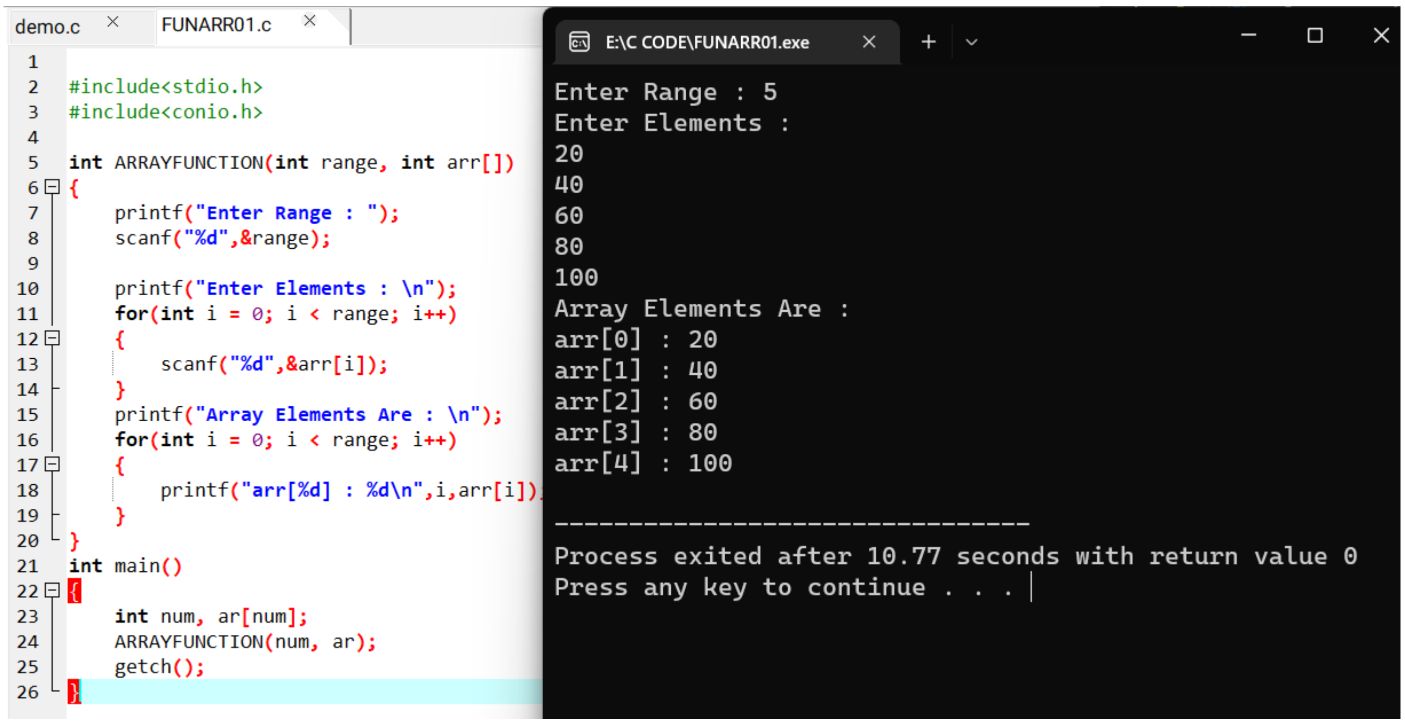
Take Nothing & Return Nothing Function & Array:



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

int ARRAYFUNCTION()
{
    int range;

    printf("Enter Range Value : ");
    scanf("%d",&range);

    int arr[range];

    printf("Enter Elements : \n");
    for(int i = 0; i < range; i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("Array Elements Are : \n");
    for(int i = 0; i < range; i++)
    {
        printf("arr[%d] : %d\n",i,arr[i]);
    }
}
int main()
{
    ARRAYFUNCTION();
    getch();
}
```

```
Enter Range Value : 5
Enter Elements :
10
20
30
40
50
Array Elements Are :
arr[0] : 10
arr[1] : 20
arr[2] : 30
arr[3] : 40
arr[4] : 50

--------------------------------
Process exited after 7.927 seconds with return value 0
Press any key to continue . . .
```

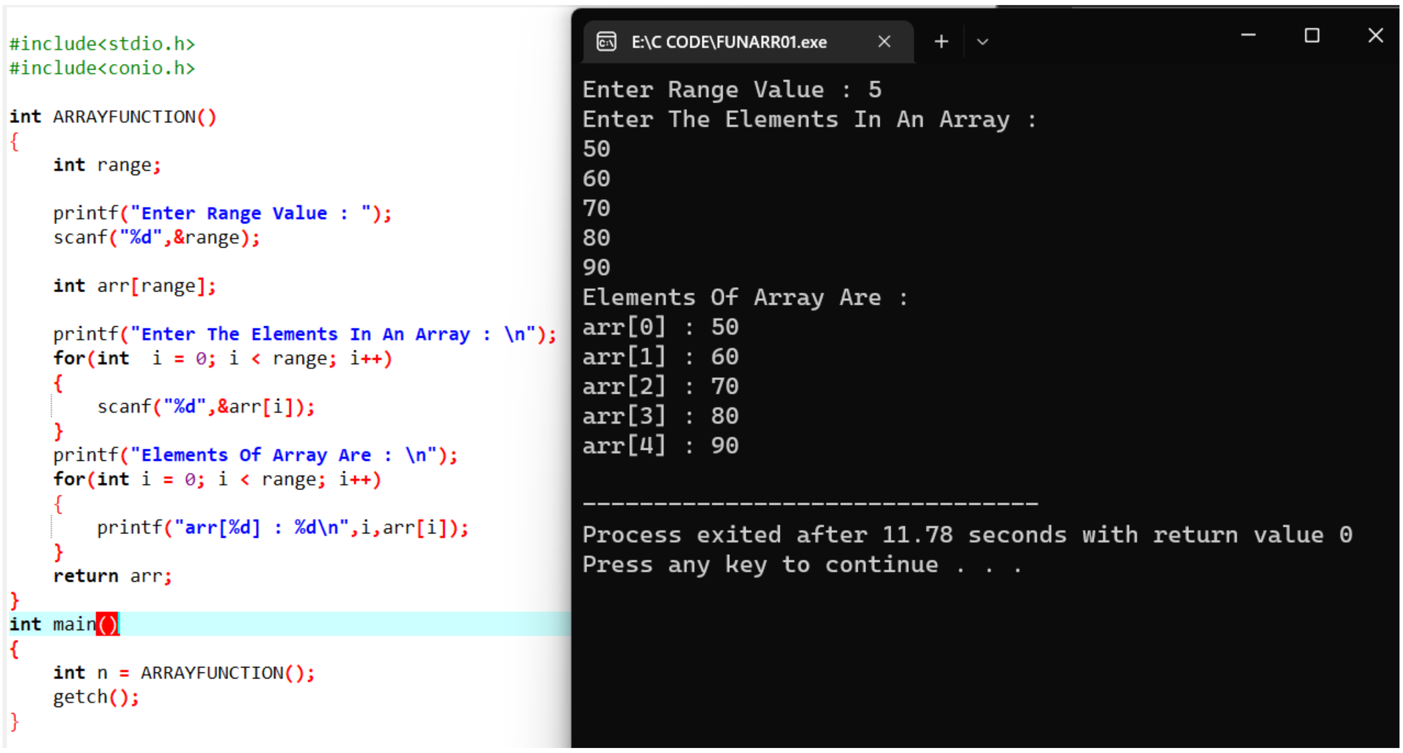
Take Something & Return Nothing Function & Array:



```c

#include<stdio.h>
#include<conio.h>

int ARRAYFUNCTION(int range, int arr[])
{
    printf("Enter Range : ");
    scanf("%d",&range);

    printf("Enter Elements : \n");
    for(int i = 0; i < range; i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("Array Elements Are : \n");
    for(int i = 0; i < range; i++)
    {
        printf("arr[%d] : %d\n",i,arr[i])
    }
}
int main()
{
    int num, ar[num];
    ARRAYFUNCTION(num, ar);
    getch();
}
```

```
Enter Range : 5
Enter Elements :
20
40
60
80
100
Array Elements Are :
arr[0] : 20
arr[1] : 40
arr[2] : 60
arr[3] : 80
arr[4] : 100

--------------------------------
Process exited after 10.77 seconds with return value 0
Press any key to continue . . .
```

Take Nothing & Return Something Function & Array:

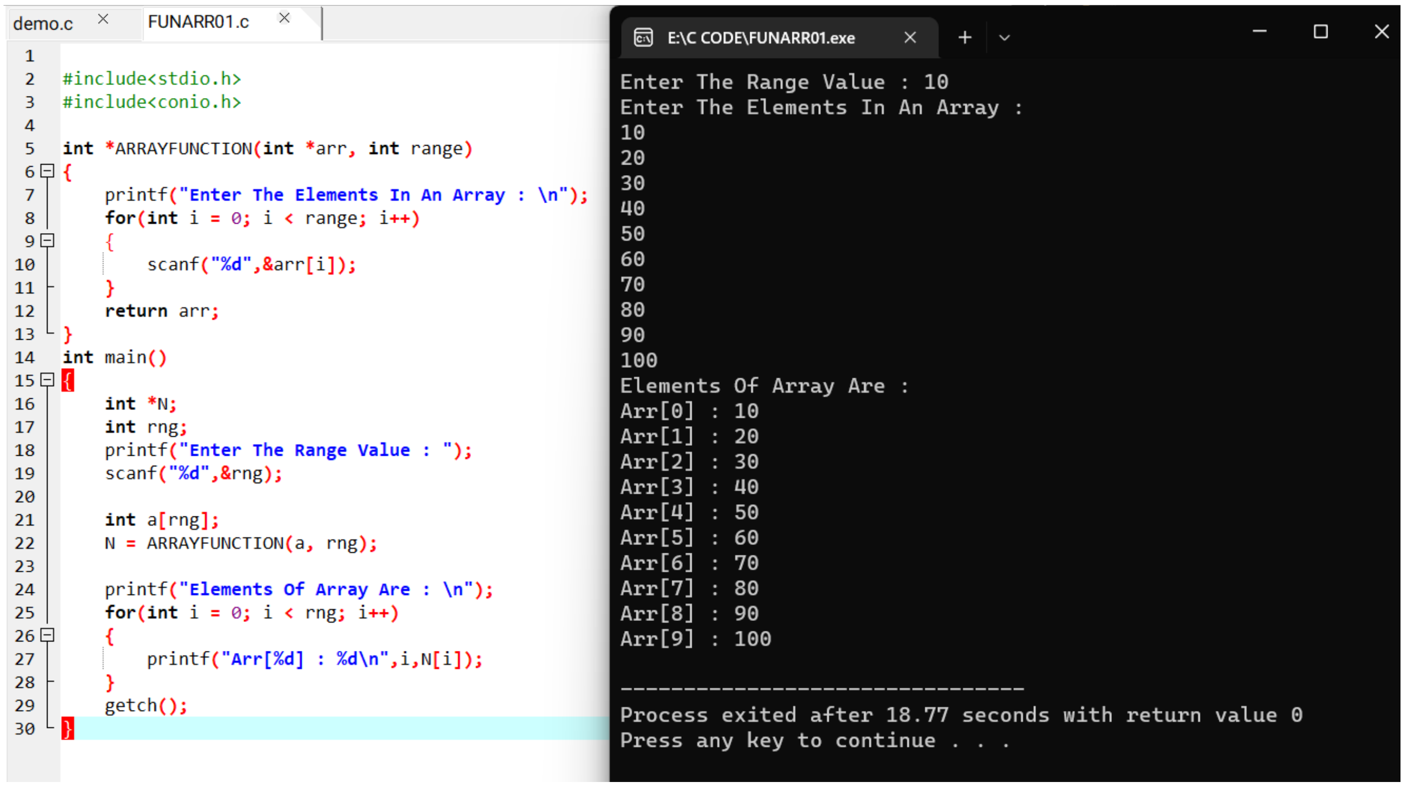


```c
#include<stdio.h>
#include<conio.h>

int ARRAYFUNCTION()
{
    int range;

    printf("Enter Range Value : ");
    scanf("%d",&range);

    int arr[range];

    printf("Enter The Elements In An Array : \n");
    for(int  i = 0; i < range; i++)
    {
        scanf("%d",&arr[i]);
    }
    printf("Elements Of Array Are : \n");
    for(int i = 0; i < range; i++)
    {
        printf("arr[%d] : %d\n",i,arr[i]);
    }
    return arr;
}
int main()
{
    int n = ARRAYFUNCTION();
    getch();
}
```

E:\C CODE\FUNARR01.exe

```
Enter Range Value : 5
Enter The Elements In An Array :
50
60
70
80
90
Elements Of Array Are :
arr[0] : 50
arr[1] : 60
arr[2] : 70
arr[3] : 80
arr[4] : 90

--------------------------------
Process exited after 11.78 seconds with return value 0
Press any key to continue . . .
```

Take Something & Return Something Function & Array:



demo.c FUNARR01.c

```c

#include<stdio.h>
#include<conio.h>

int *ARRAYFUNCTION(int *arr, int range)
{
    printf("Enter The Elements In An Array : \n");
    for(int i = 0; i < range; i++)
    {
        scanf("%d",&arr[i]);
    }
    return arr;
}
int main()
{
    int *N;
    int rng;
    printf("Enter The Range Value : ");
    scanf("%d",&rng);

    int a[rng];
    N = ARRAYFUNCTION(a, rng);

    printf("Elements Of Array Are : \n");
    for(int i = 0; i < rng; i++)
    {
        printf("Arr[%d] : %d\n",i,N[i]);
    }
    getch();
}
```

E:\C CODE\FUNARR01.exe

```
Enter The Range Value : 10
Enter The Elements In An Array :
10
20
30
40
50
60
70
80
90
100
Elements Of Array Are :
Arr[0] : 10
Arr[1] : 20
Arr[2] : 30
Arr[3] : 40
Arr[4] : 50
Arr[5] : 60
Arr[6] : 70
Arr[7] : 80
Arr[8] : 90
Arr[9] : 100

--------------------------------
Process exited after 18.77 seconds with return value 0
Press any key to continue . . .
```