Programming in ANSI C- 7th Edition- E. Balagurusamy

Presented by Teresa Jency Bala

ID-19 385 20 113

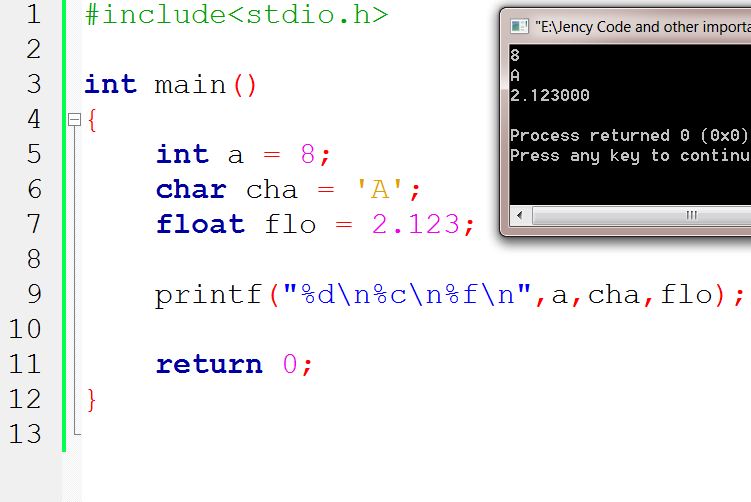
Department CSE

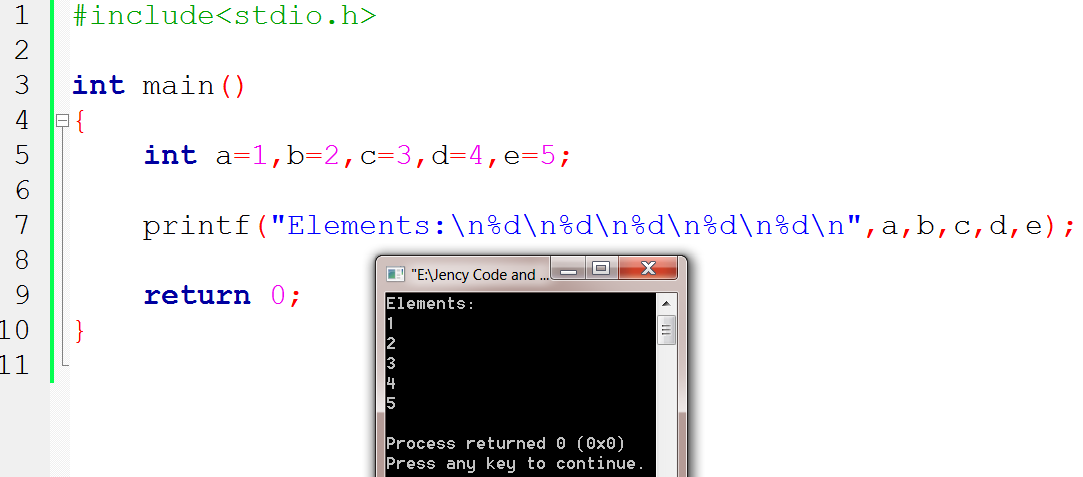
CH-7 Array

Learning objectives:

1. Define the concept of arrays
2. Determine how one-dimensional array is declared and initialized
3. Know the concept of two-dimensional arrays
4. Discuss how two-dimensional array is declared and initialized
5. Describe multi-dimensional arrays
6. Explain dynamic arrays

Normally one variable of one datatype can hold only one value



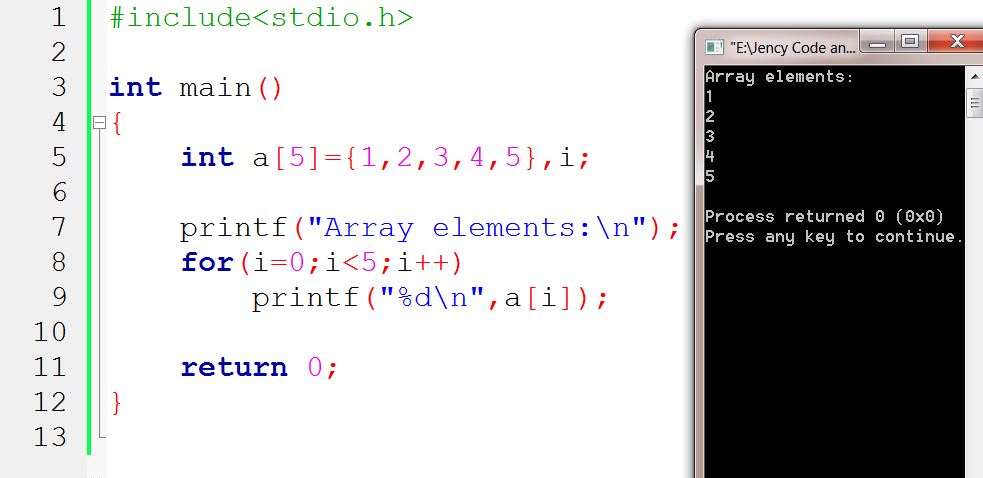
We **don’t** need to declare values for elements. Like below:

What is array?

An array is a fixed size sequenced collection of elements of the SAME datatype.

Syntax to declare array:

Datatype variable-name [size]



a[0]= 1

a[1]= 2

a[2]= 3

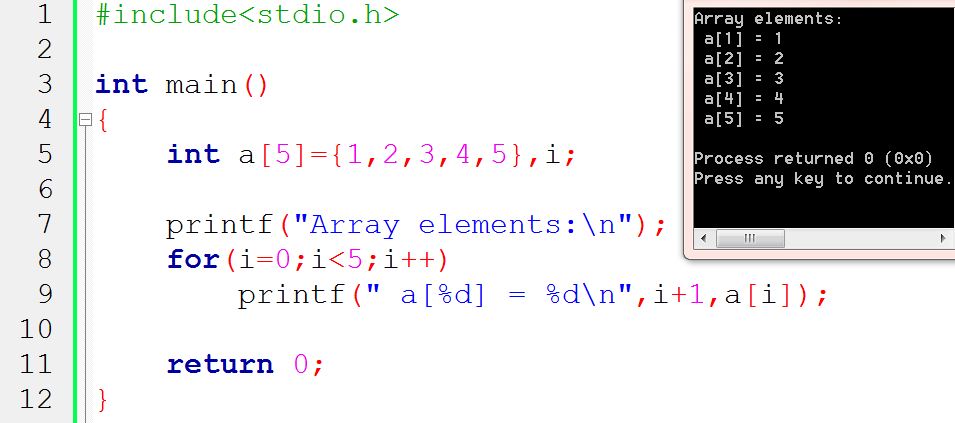
a[3]= 4

a[4]= 5

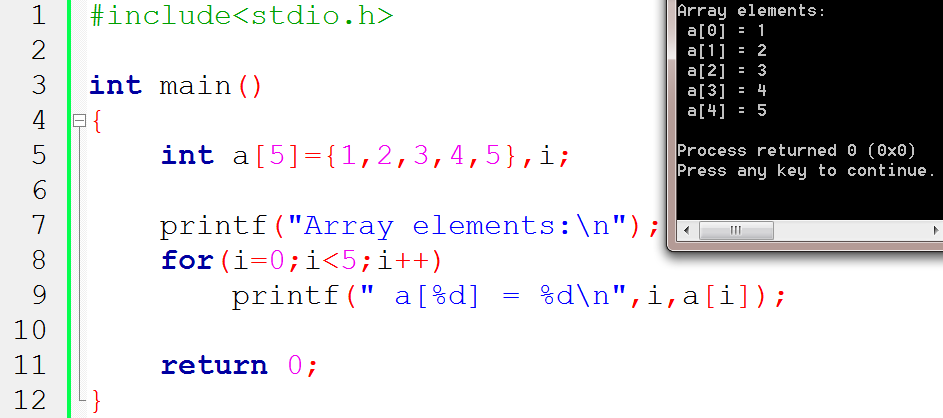
One-dimensional:

Just by declaring one variable we can put an INDEX number which is merely the number variables.

We can index starting with 0 which is preferred by computer. But we might get confused so we can state indexing with 1.



OR,



i

Compile Time Initialization:

We can initialize the elements of array in same way as the ordinary variable when they are declared.

Datatype array-name [size] = {list of values}

Int number[3] = {0, 1, 2};

We can omit the size during compile time initialization only.

int number[ ] = {1, 2, 3, 4};

This approach works fine as long as we initialize every element in the array.

Character array Initialization

char name[ ] = {‘J’,ʻe’,ʻn’,ʻc’,ʻy’,ʻ\0’};

or,

char name[ ]=ʺJency”;

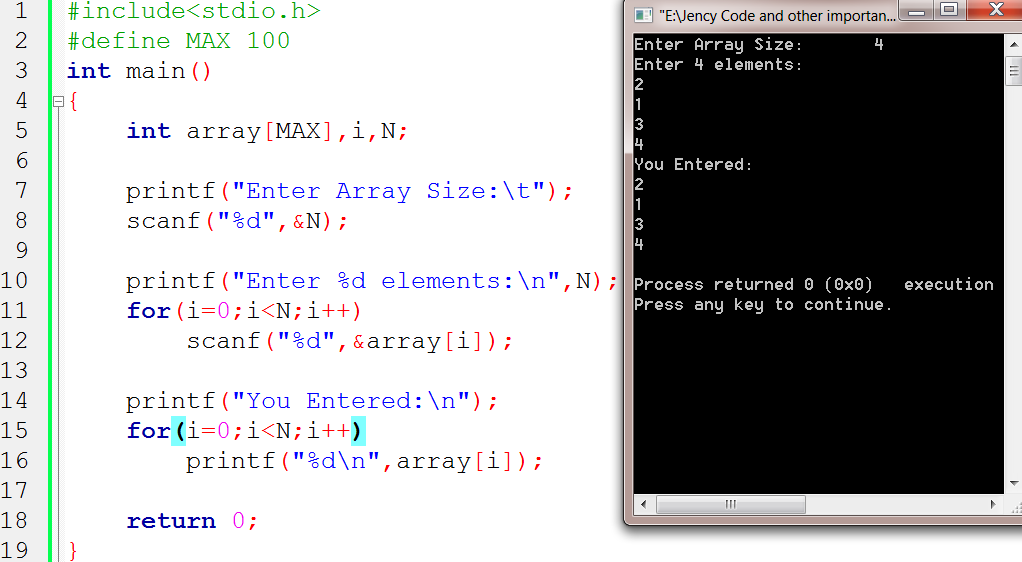
Compile time initialization may be partial.

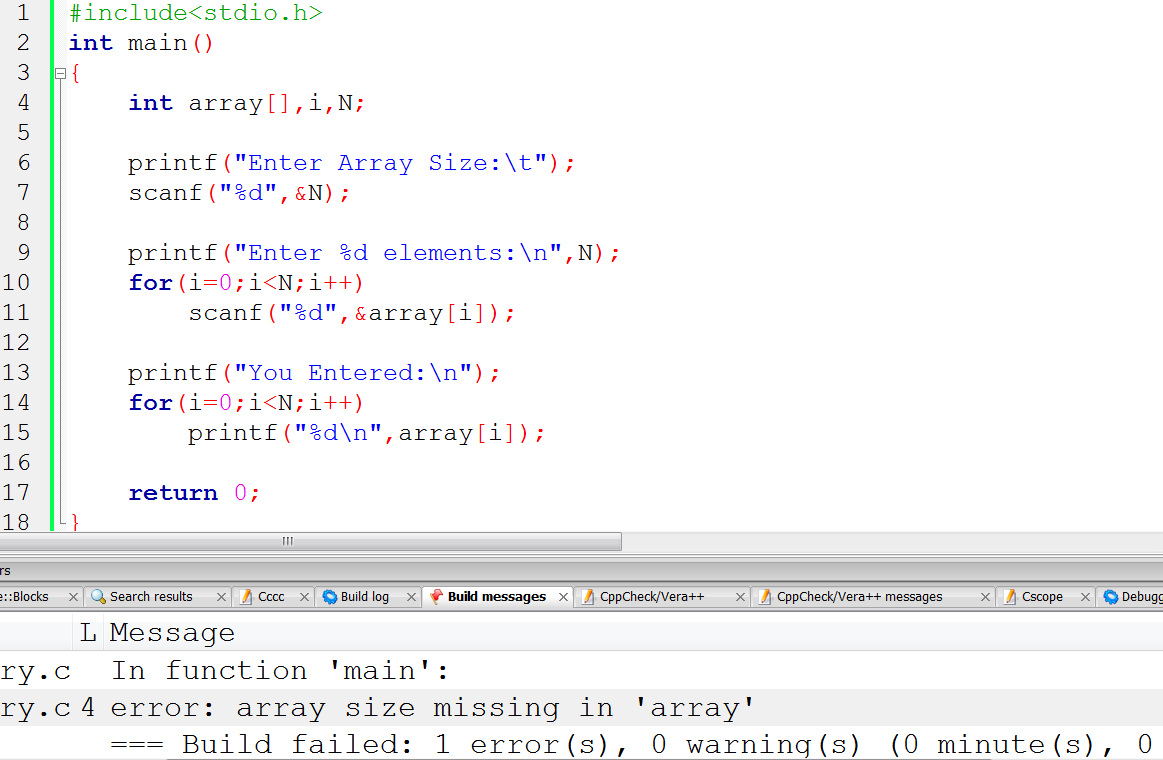
int number[5]={10, 20, 30};

Here, array index number initialized is 5 but there are 3 elements. The remaining 2 places are Zero and if the array type is char the Null.

int number[2] = {1,2,3,4}; In this case the declared size has more initialized elements. The compiler will create error.

Run time initialization:

An array can be Explicitly initialized at run time. This approach is usually applied for initializing user input data. Using for loop can help in this case.

We can’t leave size empty in array

Searching And Sorting

Sorting: The process of arranging elements in the list according to their values, in ascending or descending order. A sorted list is called an ordered list. Sorted lists are especially important in list searching because they facilitate rapid search operation. Important and simple sorting techniques:

* Shell sort
* Merge sort
* Quick sort
* Bubble sort
* Selection sort
* Insertion sort

Searching: is the process of finding the location of the specific element in a list. The specified element is often called the search key. If the search key with list element values, the search is said to be successful else unsuccessful. The most commonly used search techniques are:

* Sequential Search
* Binary Search

Two-dimensional Arrays:

We represent a particular value in matrix using 2 subscripts such as vrc  here, r is for row and c is for column.

Syntax: datatype array\_name[row\_size][column\_size];

Like single-dimensional arrays, each dimension of the array is indexed from zero to its maximum size-1; The first index selects the row and the second index selects the column within row.



2D Array Compile time initialization:

Array size declaration and values initialized in braces:

int table[2][3]={0, 0, 0, 1, 1, 1};

or,

The initialization is done row by row

int table[2][3] = {{0, 0, 0},{1, 1, 1} };

int table[2][3] = {

We can initialize a two dimensional array in the form of matrix

{0, 0, 0},

{1, 1, 1}

};

Or,

When the array is completely initialized with all values, explicitly, we need not specify the size of the dimension

int table[ ][3] = = {

{0, 0, 0},

{1, 1, 1}

};

int table[2][3] = {

{1,2},

If the values are missing in initialize, they are automatically set to Zero

{2}

};

int table[2][3] = {{0}, {0}, {0}};

When all the elements are to be initialized to Zero, this short-cut may be used

or,

int table[2][3] = {0, 0};

#include <stdio.h>

int main () {

/\* an array with 5 rows and 2 columns\*/

int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};

int i, j;

/\* output each array element's value \*/

for ( i = 0; i < 5; i++ ) {

for ( j = 0; j < 2; j++ ) {

printf("a[%d][%d] = %d\n", i,j, a[i][j] );

}

}

return 0;

}

a[0][0]: 0

a[0][1]: 0

a[1][0]: 1

a[1][1]: 2

a[2][0]: 2

a[2][1]: 4

a[3][0]: 3

a[3][1]: 6

a[4][0]: 4

a[4][1]: 8

2D matrix Run time Input and Display:

#include<stdio.h>

#define MAX 10

int main()

{

int array[MAX][MAX],i, j, r, c;

printf("Enter row and column number:\n");

scanf("%d %d", &r, &c);

printf("Enter %d X %d elements:\n", r, c);

for(i = 0; i <r; i++)

{

for(j=0;j<c; j++)

{

printf("Enter array[%d][%d]: ",i+1,j+1);

scanf("%d", &array[i][j]);

}

}

printf("Your entered 2D matrix of %dX%d elements:\n", r, c);

for(i=0;i<r; i++)

{

for(j=0;j<c; j++)

{

printf("%5d", array[i][j]);

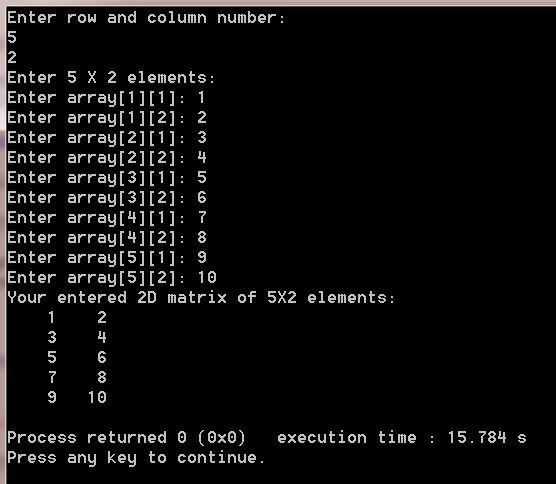
}

printf("\n");

}

return 0;

}



Multi-dimensional arrays:

C allows three or more dimensions. The exact limit is determined by the compiler. It's an array or collection of 2D arrays, and a 2D array is an array of 1D array.

The general form of a multi-dimensional array is

datatype arrary\_name[s1][s2]…..[si]; here si is size of i-th dimension.

Examples:

int survey[3][5][12]; holds: 3\*5\*12=180 integer type elements

Declaration and Initialization 3D Array

#include<stdio.h>

int main()

{

int i, j, k;

int arr[3][3][3]=

{

{

{11, 12, 13},

{14, 15, 16},

{17, 18, 19}

},

{

{21, 22, 23},

{24, 25, 26},

{27, 28, 29}

},

{

{31, 32, 33},

{34, 35, 36},

{37, 38, 39}

},

};

printf(":::3D Array Elements:::\n\n");

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

for(k=0;k<3;k++)

{

printf("%d\t", arr[i][j][k]);

}

printf("\n");

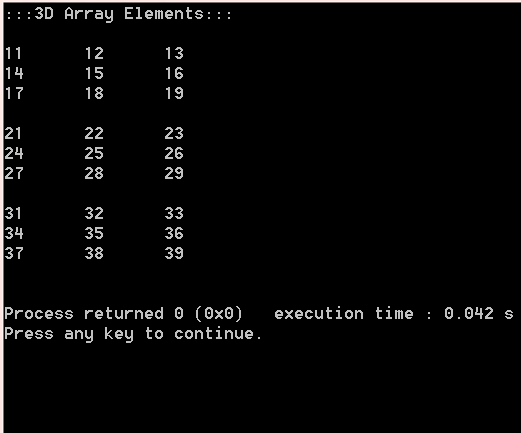
}

printf("\n");

}

return 0;

}



Dynamic Array:

We create arrays at compile time. An array created at compile time by specifying SIZE in the source code has a fixed size and cannot be modified at run time. The process of allocating memory at compile time is known as Static Memory Allocation.

Considering a situation where we want to use array that can vary greatly in size. In C it is possible to allocate memory to array at run time are called Dynamic arrays.

Dynamic arrays are created using what are known as pointer variables and memory management function malloc, calloc and realloc. These functions are included in header file <stdlib.h>. These are used in data structure such as linked lists, stacks and queues.