Programing Methodology in C

Lecture 8 – Arrays in C Programing Language

By Elubu Joseph

josebulinga@gmail.com

Agenda

1. Arrays
    i. Declaration
    ii. Initialization and
    iii. Types of arrays
2. String and Character Array

Arrays in C Programing Language

Definition.

In Mathematics, array is an arrangement of quantities or symbols in rows and columns; a matrix.

In Computing an array is ordered set of related elements.

or

An array is a finite ordered collection of homogenous data, stored in continuous memory locations. In C language, arrays are referred to as structured data types.

Arrays in C Programing Language +

Here the word: ,

- **finite** *means* data range must be defined.
- **ordered** *means* data must be stored in continuous memory addresses.
- **homogenous** means data must be of similar data type.

Where are arrays used?

Arrays are used to: -

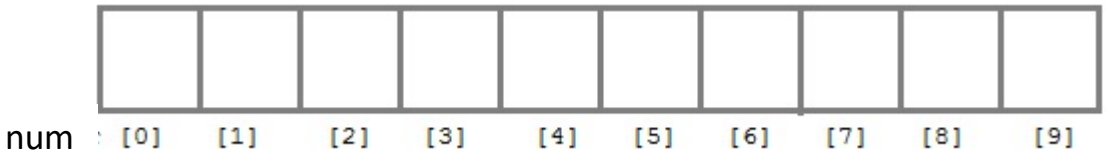
- store list of Employee or Student names,
- store marks of students,
- or to store list of numbers or characters etc.

Since arrays provide an easy way to represent data, it is classified amongst the data structures in C. Other data structures in C are **structure**, **lists**, **queues**, **trees** etc. Array can be used to represent not only simple list of data but also table of data in two or three dimensions.

Array Declaration

Like any other variable, arrays must be declared before they are used. General form of array declaration is,

```
data-type variable-name[size];
/* Example of array declaration */
int num[10];
```



Here int is the data type, num is the name of the array and 10 is the size of array. It means array num can only contain 10 elements of int type.

Index of an array starts from **0** to **size-1** i.e first element of num array will be stored at num[0] address and the last element will occupy num[9].

Array Declaration+

Array helps you to store multiple values in a single variable. In the example below three arrays are declared;- **joe**, **name** and **surname**.

```
Int joe[6];  This will enable you store 6 values from index 0 to index 5. or
char name[25]; will enable you store 25 values from index 0 to index 24.
Or
char surname[]; will store un known number of values
```

Initialization of an Array

After an array is declared it must be initialized. Otherwise, it will contain **garbage** value(any random value). An array can be initialized at either **compile time** or at **runtime**.

Compile time Array initialization

Compile time initialization of array elements is same as ordinary variable initialization. The general form of initialization of array is,

```
ata-type array-name[size] = { list of values };
/* Here are a few examples */
int marks[4]={ 67, 87, 56, 77 }; // integer array initialization
float area[5]={ 23.4, 6.8, 5.5 }; // float array initialization
int marks[4]={ 67, 87, 56, 77, 59 }; Invalid array initialization
```

Initialization of an Array

E.g. you can assign values to each index as below.

```
char name[7];
 name[0] = ‘B’;
 name[1]= ‘a’;
 name[2] = ‘p’;
 name[3] = ’t’;
 name[4] = ’i’;
 name[5] = ’s’;
name[5] = ’s’;
```

You can also initialize your array variable as: -

```
char name []= {‘B’,’a’,’p’,’t’,’i’,’s’,’t’};
```

This is much better because it dose not limit the number of items in name[] array.

Initialization of an Array

E.g. you can assign values to each index as below.

```
int joe[6];
        joe[0] = 10;
        joe[1]= 20;
        joe[2] = 30;
        joe[3] = 40;
        joe[4] = 50;
        joe[5] = 60;
```

Meaning that in each index there is a value.

Question?

1. If the value at index 0 of the name array below is B,

```
char name[7];
 name[0] = ‘B’;
 name[1]= ‘a’;
 name[2] = ‘p’;
 name[3] = ’t’;
 name[4] = ’i’;
 name[5] = ’s’;
name[6] = ’t’
```

2. What is the output of the following line of code?

printf(“%c”, name[3]);

Important!

Live the array box empty and list your items directly to avoid limitations

```
char name []= {‘B’,’a’,’p’,’t’,’i’,’s’,’t’};
```

Note that ‘B’ above is at index 0 and ‘a’ is at index 1 >>>> last ‘t’ is at index 6.

Sample program that prints out specific array elements

```
#include<stdio.h>
int main(){
int joe[6] = {10, 20,30,40,50,60}
        printf("%d ======%d", joe[4], joe[1]);
}
```

What will be the out put of the above code?

50======20

Sample program that prints out the elements of an array initialized at compile time.

```
#include<stdio.h> void main() {
int i; int num[] = {2, 3, 4};
// Compile time array initialization
for(i = 0 ; i < 3 ; i++) {
printf("%d\t",num[i]);
}
}
```

OUTPUT

2 3 4

Runtime Array initialization

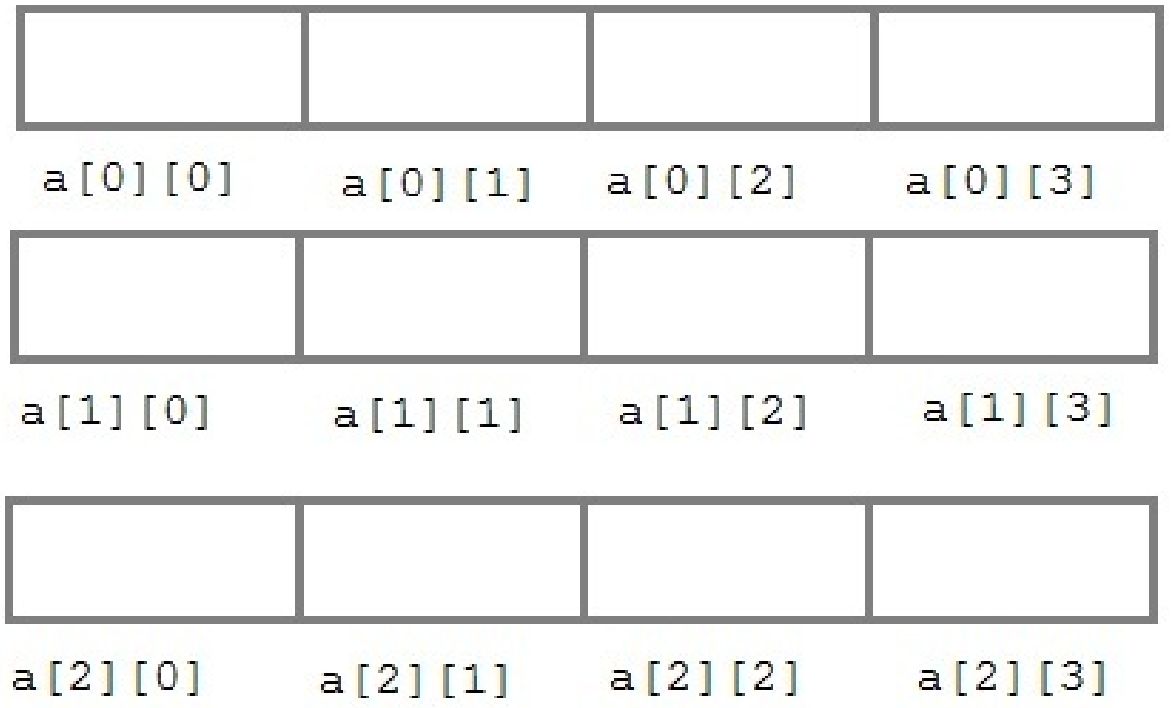
An array can also be initialized at runtime using scanf() function. This approach is usually used for initializing large arrays, or to initialize arrays with user specified values.

Example,

```
#include<stdio.h>
void main() {
int num[4]; int i, j;
printf("Enter array element");
for(i = 0; i < 4; i++) {
scanf("%d", &num[i]);
//Run time array initialization
}
for(j = 0; j < 4; j++) {
printf("%d\n", num[j]);
}
}
```

Two dimensional Arrays

C language supports multidimensional arrays also. The simplest form of a multidimensional array is the two-dimensional array. Both the row's and column's index begins from 0. Two-dimensional arrays are declared as follows,




```
data-type array-name[row-size][column-size]
/* Example */
int a[3][4];
```

A two dimensional array can also be declared and initialized at the same time. For example,

```
int num[][3] = { {0,0,0}, {1,1,1} };
```

Note: We have not assigned any row value to our array in the above example. It means we can initialize any number of rows. But, we must always specify number of columns, else it will give a compile time error. Here, a 2*3 multi-dimensional matrix is created.

Runtime initialization of a two dimensional Array

```
#include<stdio.h>
void main() {
int num[3][4]; int i, j, k;
printf("Enter array element \n");
for(i = 0; i < 3;i++) {
      for(j = 0; j < 4; j++) {
      scanf("%d", &num[i][j]);
      }
}
printf("You entered\t");
for(i = 0; i < 3; i++) {
      for(j = 0; j < 4; j++) {
      printf(“You entered %d\t", num[i][j]);
} } }
```

String and Character Array

String is a sequence of characters that is treated as a single data item and terminated by null character '\0'. Remember that C language does not support strings as a data type. A **string** is actually one-dimensional array of characters in C language. These are often used to create meaningful and readable programs.

For example:

The string "hello world" contains 12 characters including '\0' character which is automatically added by the compiler at the end of the string.

Declaring and Initializing a string array

There are different ways to initialize a character array variable.

```
char name[15] = “Kumi_University"; // valid character array initialization
char name[10] = {'L','e','s','s','o','n','s','\0'}; // valid initialization
```

String Input and Output

Input function scanf() can be used with **%s** format specifier to read a string input from the terminal. But there is one problem with scanf() function, it terminates its input on the first white space it encounters.

Therefore if you try to read an input string "Hello World" using scanf() function, it will only read **Hello** and terminate after encountering white spaces.

However, C supports a format specification known as the **edit set conversion code %[..]** that can be used to read a line containing a variety of characters, including white spaces.

Using the **edit set conversion code %[..]**

```
#include<stdio.h>
#include<string.h>
void main() {
     char str[20];
     printf("Enter a string"); scanf("%[^\n]", &str);
     //scanning the whole string, including the white
spaces
     printf("%s", str);
}
```

Using gets()function

Another method to read character string with white spaces from terminal is by using the gets()function.

```
char text[20];
 gets(text);
printf("%s", text);
```

String Character and Array

The best way of handing strings is by using arrays. And arrays can do this perfectly well when we involve the use of string handling functions in c.

C language supports a large number of **string handling functions** that can be used to carry out many of the string manipulations.

These functions are packaged in **string.h** library. Hence, you must include **string.h** header file in your programs to use these functions.

More String Handling Functions

The following are the most commonly used string handling functions.

Method	Description
strcat()	It is used to concatenate(combine) two strings
strlen()	It is used to show length of a string
strrev()	It is used to show reverse of a string
strcpy()	Copies one string into another
strcmp()	It is used to compare two string

1. strcat() function

strcat() function will add the string **“Osi"** to **“wache"** i.e it will ouput Osiwache.

```
strcat(“Osi", “wache");
```

Output: osiwache

2. strlen() function

strlen() function will return the length of the string passed to it.

```
int j; j = strlen(“Kumiuniversity");
printf("%d",j);
```

Output
14

3. strcpy() function

Copies the second string argument to the first string argument.

```
#include<stdio.h> #include<string.h>
int main() {
char s1[50]; char s2[50];
strcpy(s1, “LearnPrograming"); //copies
“LearnPrograming" to string s1
strcpy(s2, s1); //copies string s1 to string s2
printf("%s\n", s2);
return(0);
}
```

Output : LearnPrograming

4. strrev() function

It is used to reverse the given string expression.

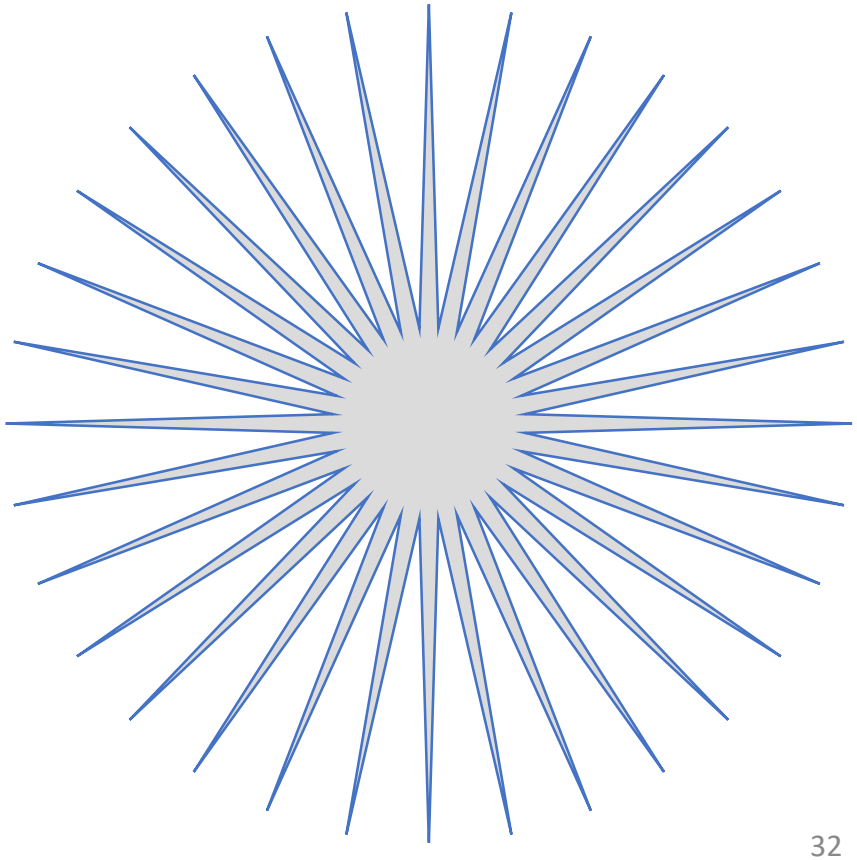
```
#include<stdio.h>
int main() {
char s1[50];
printf("Enter your full name ");
gets(s1);
printf("\nYour reverse name is:%s",strrev(s1));
return(0);
}
```

Output
Enter your string: Your reverse name is: ……..

Summary

1. Arrays
    i. Declaration
    ii. Initialization
    iii. Types of arrays
2. String and Character Array(looked at functions such as strcmp(), strcpy(), strlen(), strrev() and strcat())

Thank you for you attention



Reference

Arrays in C. Studytonight.com. (n.d.). Retrieved November 11, 2021, from https://www.studytonight.com/c/arrays-in-c.php.

String and character array. Studytonight.com. (n.d.). Retrieved November 11, 2021, from https://www.studytonight.com/c/string-and-character-array.php.