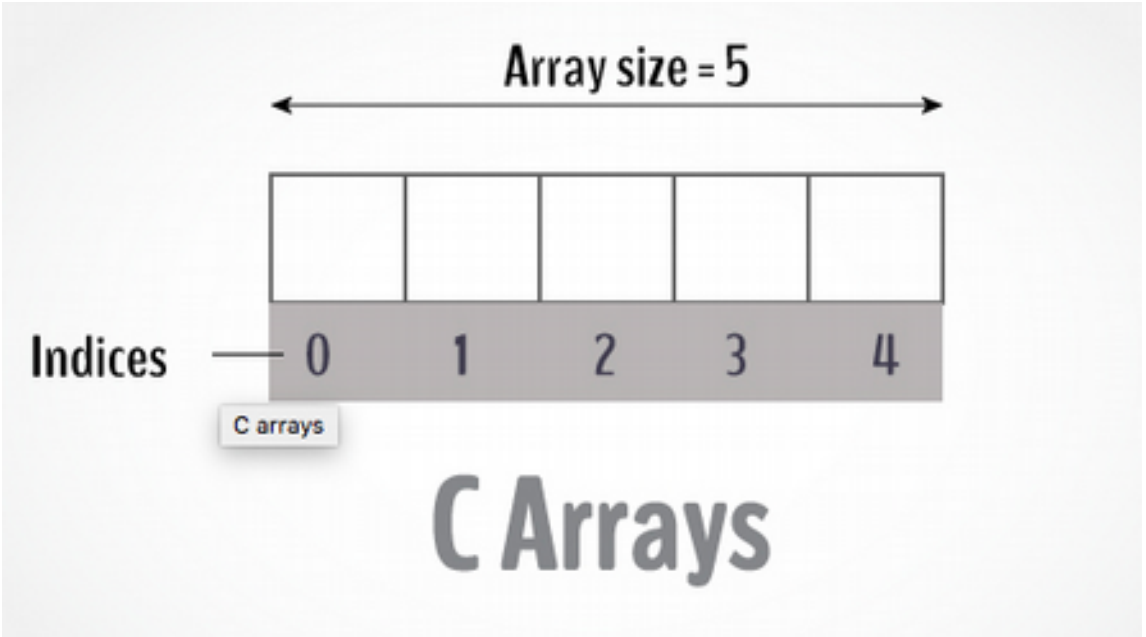
C Programming

Kasun De Zoysa



An array is a collection of data that holds fixed number of values of same type.

For example: if you want to store marks of 100 students, you can create an array for it.

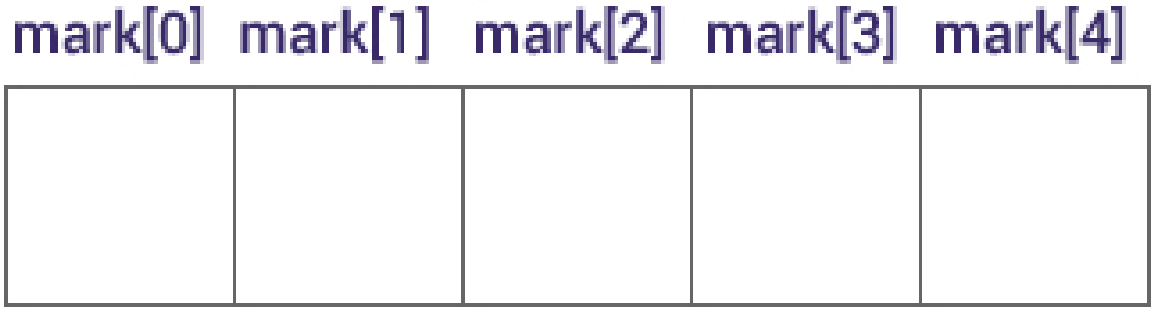
Arrays are of two types:
1. One-dimensional arrays
2. Multidimensional arrays

How to declare an array in C?

```
data_type array_name[array_size];
```

For example,

```
float mark[5];
```



How to access them?

You can access elements of an array by indices. Suppose you declared an array mark.

Then the first element is mark[0], second element is mark[1] and so on.

Few key notes:
Arrays have 0 as the first index not 1.
In this example, mark[0]

If the size of an array is n, to access the last element, (n-1) index is used. In this example, mark[4]

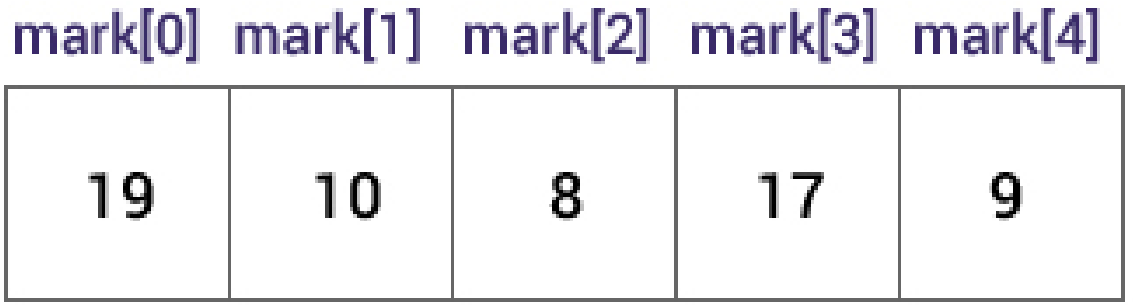
Suppose the starting address of mark[0] is 2120d. Then, the next address, a[1], will be 2124d, address of a[2] will be 2128d and so on. It's because the size of a float is 4 bytes.

How to initialize an array

```
int mark[5] = {19, 10, 8, 17, 9};
```

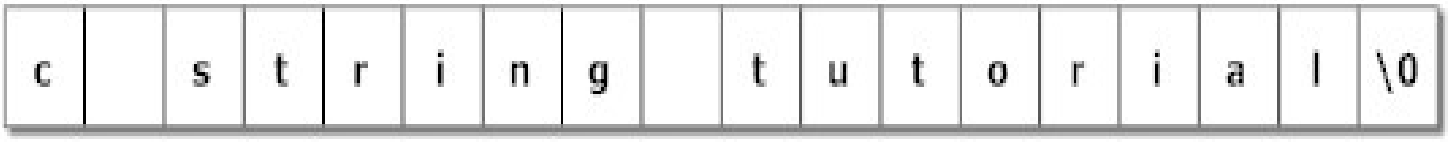
Another method to initialize array during declaration:

```
int mark[] = {19, 10, 8, 17, 9};
```



In C programming, array of characters is called a string. A string is terminated by a null character /0.

For example:

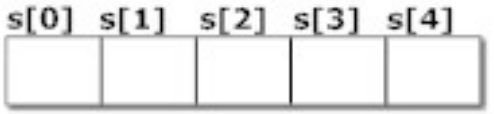


Declaration of strings

Strings are declared in a similar manner as arrays. Only difference is that, strings are of char type.

Using arrays

```
char s[5];
```

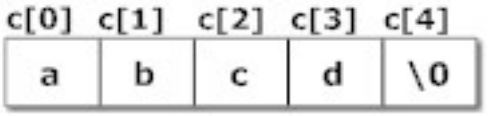


Initialization of strings

In C, string can be initialized in a number of different ways.

Using arrays

```
char c[] = "abcd";
      OR,
char c[50] = "abcd";
      OR,
char c[] = {'a', 'b', 'c', 'd', '\0'};
      OR,
char c[5] = {'a', 'b', 'c', 'd', '\0'};
```



Reading Strings from user

```c
#include <stdio.h>
int main()
{
      char name[20];
      printf("Enter name: ");
      scanf("%s", name);
      printf("Your name is %s.", name);
      return 0;
}
```

```c
#include <stdio.h>
int main()
{
      char name[30];
      printf("Enter name: ");
      gets(name);        //Function to read string from user.
      printf("Name: ");
      puts(name);      //Function to display string.
      return 0;
}
```

String manipulation

Function	Work of Function
strlen()	Calculates the length of string
strcpy()	Copies a string to another string
strcat()	Concatenates(joins) two strings
strcmp()	Compares two string
strlwr()	Converts string to lowercase
strupr()	Converts string to uppercase

Strings handling functions are defined under "**string.h**" header file.

Command Line Arguments in C

It is possible to pass some values from the command line to your C programs when they are executed. These values are called command line arguments and many times they are important for your program especially when you want to control your program from outside instead of hard coding those values inside the code.

To pass command line arguments, we typically define main() with two arguments :
first argument is the number of command line arguments and second is list of command-line arguments.

```
int main(int argc, char *argv[]) { /* ... */ }
```

argc and *argv[]

argc (ARGument Count) is int and stores number of command-line arguments passed by the user including the name of the program.

So if we pass a value to a program, value of **argc** would be 2 (one for argument and one for program name)

argv(ARGument Vector) is array of character pointers listing all the arguments.

argv[0] is the name of the program , After that till argv[argc-1] every element is command line arguments.

```
#include <stdio.h>

int main(int argc,char *argv[]){
 printf("Number of parameters = %d \n",argc);
 for(int i=0;i<argc;i++) puts(argv[i]);
 return 0;
}
Kasuns-MacBook-Air:lec7 kasundezoysa$ ./a.out
Number of parameters = 1
./a.out
Kasuns-MacBook-Air:lec7 kasundezoysa$ ./a.out Hello
Number of parameters = 2
./a.out
Hello
Kasuns-MacBook-Air:lec7 kasundezoysa$ ./a.out Hello Kasun
Number of parameters = 3
./a.out
Hello
Kasun
Kasuns-MacBook-Air:lec7 kasundezoysa$
```

Properties of Command Line Arguments

1. They are passed to main() function.

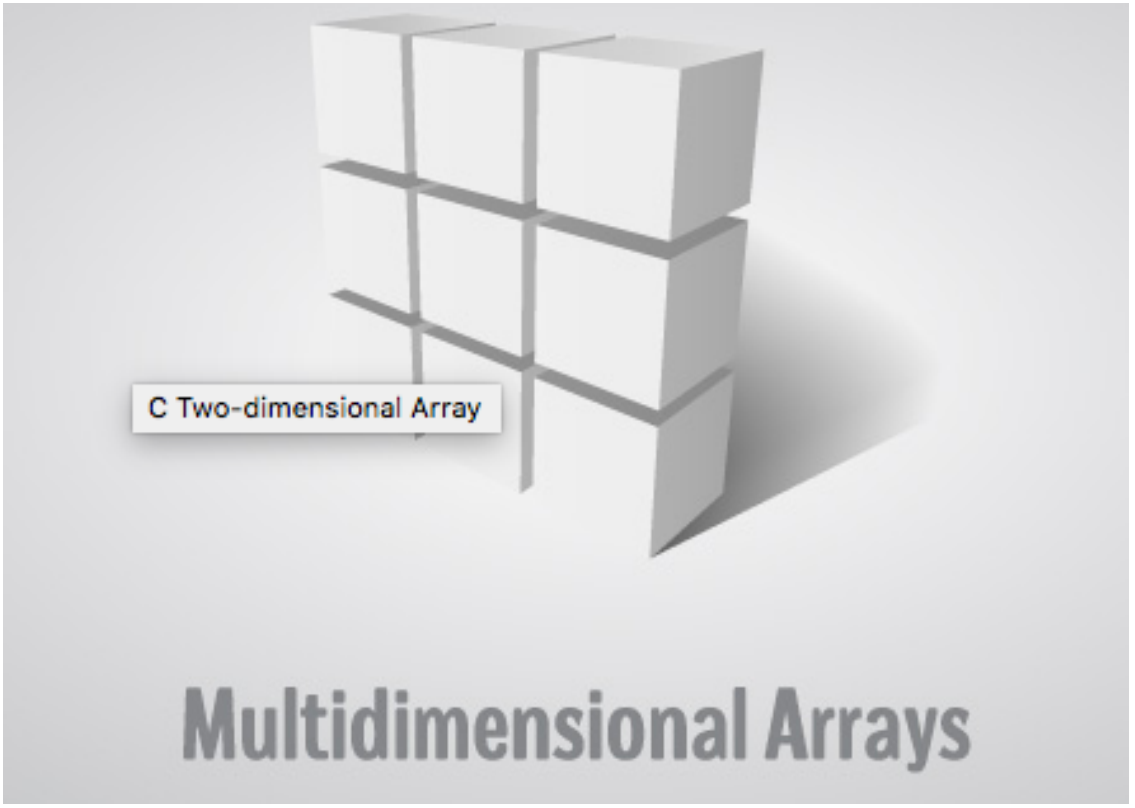
2. They are parameters/arguments supplied to the program when it is invoked.

3. They are used to control program from outside instead of hard coding those values inside the code.

4. argv[argc] is a NULL pointer.

5. argv[0] holds the name of the program.

6. argv[1] points to the first command line argument and argv[n] points last argument.



In C programming, you can create an array of arrays known as multidimensional array.

For example,

```
float x[3][4];
```

Two-dimensional (2d) array

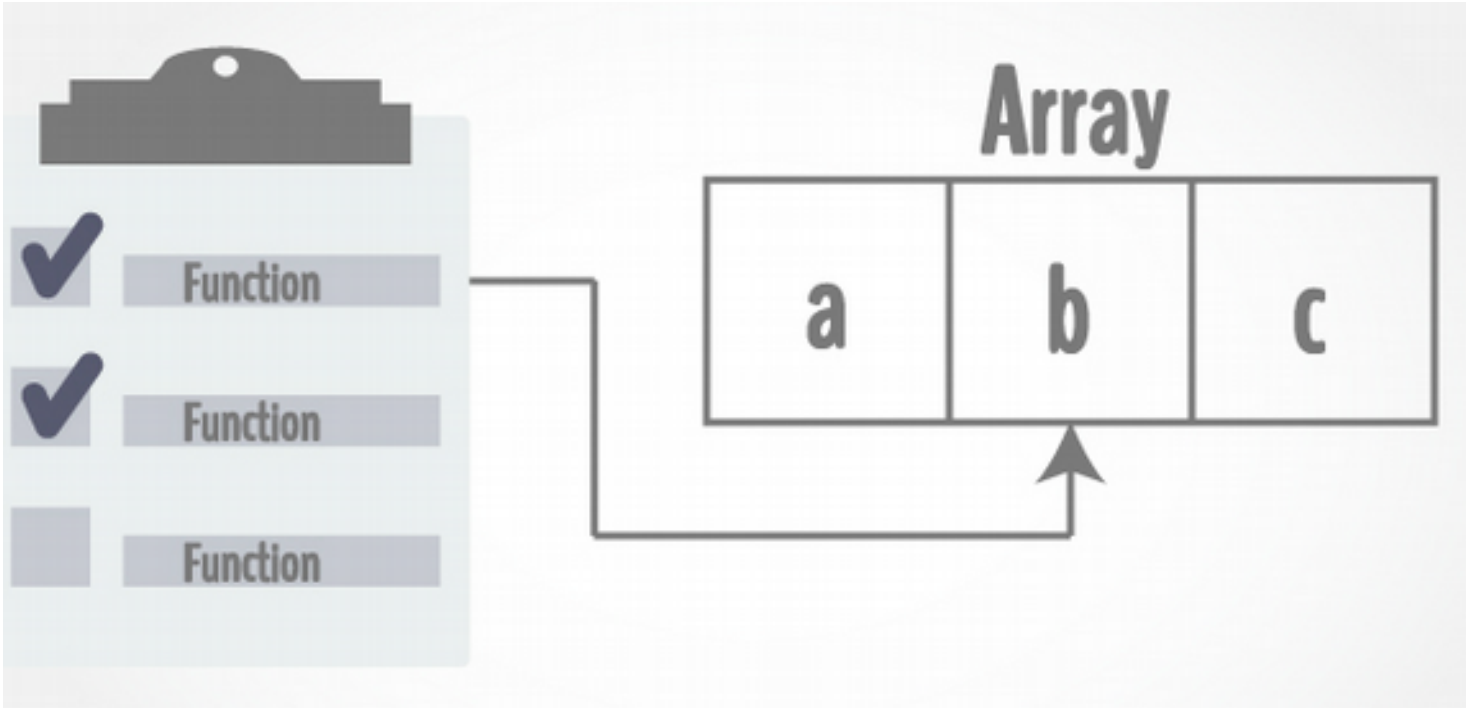
Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as table with 3 row and each row has 4 column.

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Similarly, you can declare a three-dimensional (3d) array. For example,

```
float y[2][4][3];
```

Passing One-dimensional Array in Function



While passing arrays as arguments to the function, only the name of the array is passed (,i.e, starting address of memory area is passed as argument).

Matrix

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$$

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \times \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$$

Problems?

1. Write a C program to reverse a sentence entered by user.

2. Write a program to find the Frequency of a given character.

Good Luck!