What is an Array of Strings in C?

In C, an array represents a linear collection of data elements of a similar type, thus an array of strings means a collection of several strings. The string is a one-dimensional array of characters terminated with the null character. Since strings are arrays of characters, the array of strings will evolve into a two-dimensional array of characters.

Syntax

It is possible to define an array of strings in various ways, but generally, we can do so as follows:

```
char variableName[numberOfStrings][maxSizeOfString];
```

Initialization of Array of Strings

Initialization in C means to provide the data to the variable at the time of declaration. It is pretty straightforward to initialize an array of strings. We can just create an array of some size and then we can place comma-separated strings inside that. See the example below to get a good idea about it.

```
char car[5][20] = {"Mahindra", "Audi", "Lamborghini", "Hyundai", "Tata Motors"};
```

Explanation:

In this example, we have initialized a two-dimensional character array **car** which can have a maximum of 5 strings of size 20.

Strings as an Array of Characters

As we know, C strings are nothing but an array of characters. Therefore we can initialize the string as an array of characters instead of directly writing the string in double quotes. Both approaches are the same. In the latter scenario, C interprets the string in a similar manner and appends a null character at the end.

```
char str[5] = {'H', 'e', 'l', 'l', 'o', '\0'};
char str[5] = "Hello";
```

We can also create our array of strings with this approach of writing strings as an array of characters. Let's re-write the same initialization,

```
char car[5][20] = {
    {'M','a','h','i','n','d','r','a', '\0'},
    {'A','u','d','i','\0'},
    {'L','a','m','b','h','o','r','g','i','n','i','\0'},
};
```

Example of Array of String in C

Reading and Displaying Array of Strings

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

int main()
{
    char car[5][20] = {"Mahindra", "Audi", "Hyundai", "Tata Motors"};

    for(int itemNumber = 0 ; itemNumber < 5 ; itemNumber++)
    {   //Iterate over selected string
        for(int characterPosition = 0 ; characterPosition <
strlen(car[itemNumber]) ; characterPosition++)
            printf("%c", car[itemNumber][characterPosition]);

        printf("\n");
    }

    return 0;
}
```

Output:

```
Mahindra
Audi
Lamborghini
Hyundai
Tata Motors
```

Explanation:

- We have created an array of strings, which has a fixed size of 5, and we have initialized that array at the time of declaration with the strings.
- Subsequently, we are looping in a nested manner, in which the outer loop is accessing the item of the array and the inner loop is printing the characters of that string till the end.

Some Invalid Operation on an Array of String

1. If we access a memory block that doesn't exist or is out of limits it will contain some garbage, so we should always access the memory according to its limit. All those operations in which we access invalid memory will be referred to as invalid operations in an array of strings.

2. The strings pointed by a pointer(in array of string having type char*) are internally translated as **constant char array** or read-only, so you cannot change them that's why all operations which try to modify the string by its pointer will be invalid.

An important point to note, that might create confusion is, You cannot modify a constant string but still you can always change the pointer value to point to another string.

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

int main()
{
    char* arr[2] = {"abc", "def"};
    strcpy(arr[0], "ghi"); // valid operation is arr[0]="ghi"
    printf("%s", arr[0]);
    return 0;
}
```

Output:

```
Segmentation fault (core dumped)
```

Apart from the strcpy, the use of functions like strcat, gets, fgets, and scanf to modify these types of strings will also be invalid.

3. When we write an array of strings with 2D array notation, the pointer becomes constant. So we cannot change it further to point to some other string.

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

int main()
{
    char arr[2][20] = {"abc", "def"};
    arr[0] = "ghi";

    printf("%s", arr[0]);
    return 0;
}
```

Output:

```
code.c: In function 'main':
code.c:7:12: error: assignment to expression with array type
    7 |     arr[0] = "ghi";
      |            ^
```

The above code gives an error as the name of the array is a constant pointer and we try to change it. But we can change the content of that string.

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

int main()
{
    char arr[2][20] = {"abc", "def"};

    // arr[0] = "ghi";  Invalid
    strcpy(arr[0], "ghi");
    printf("%s", arr[0]);
    return 0;
}
```

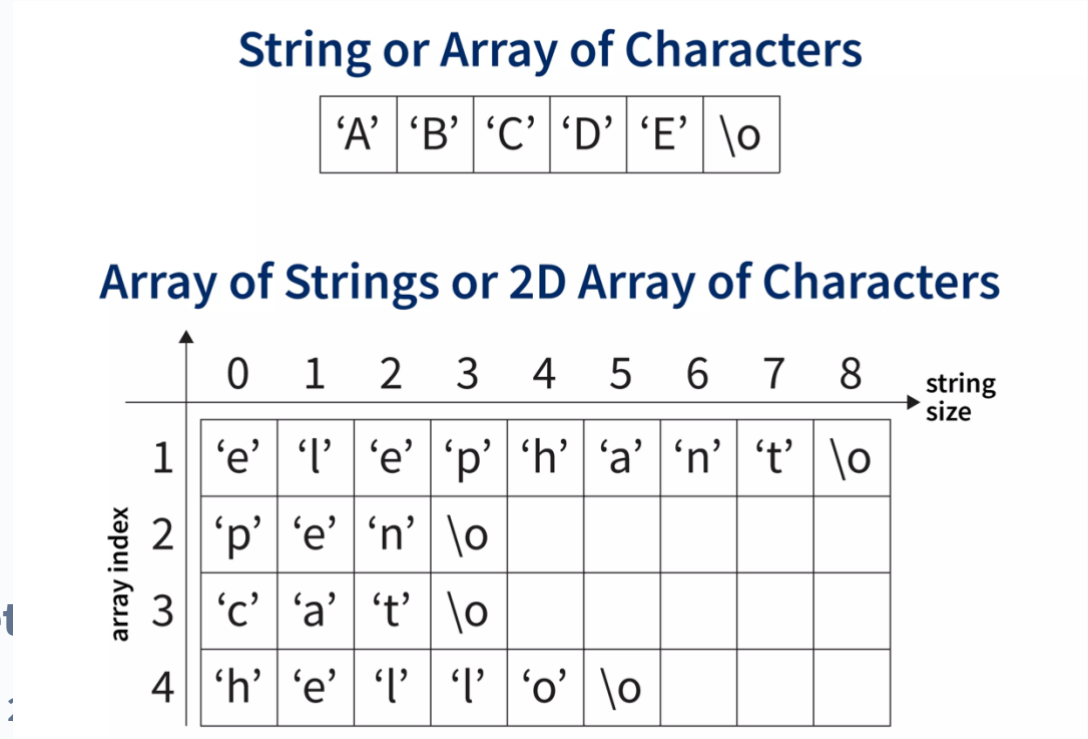
Output:

```
ghi
```

In the above code, we change the content of the string pointer by arr[0] which is valid as it is still pointing to the same location.

How Does Array of Strings Work in C?

Strings in C are a collection of characters, more formally arrays of characters that are terminated by a null character, which mean the last character will be \0 to determine that it is the end of the character array. So, the array of strings works in a similar way. It is just a two-dimensional character array.



Met

Use

As we know strings are nothing but an array of characters and hence the array of a string would be a two-dimensional array of characters. Suffice it to say, we can use a 2D character array to declare an array of strings in C.

```
char car[5][20] = {
    {'M','a','h','i','n','d','r','a', '\0'},
    {'A','u','d','i','\0'},
    {'L','a','m','b','h','o','r','g','i','n','i','\0'},
    {'H','y','u','n','d','a','i','\0'},
    {'T','a','t','a', ' ', 'M', 'o', 't', 'o', 'r','s','\0'}
};
```

Use the char* Array Notation to Declare Array of Strings in C

Before discussing this approach, let's understand what * means. When we write the * in front of the data type it becomes the pointer, which can point to the memory block of the same data type. Now if we assign the array to the pointer it points to the base address of the array. This means we can store the base address of an array in the pointer and then traverse the entire array. So char* creates a pointer that points to the base address of the string. To declare an array of strings we can create an array of these kinds of pointers. See the example below for a better illustration.

An important point to note is, that internally this **pointer to string** notation is translated as **const** character array, which means you cannot modify them.

```
#include <stdio.h>

int main()
{
    char *car[5] = {"Mahindra", "Audi", "Lambhorgini", "Hyundai", "Tata
Motors"};

    for(int i = 0 ; i < 5 ; i++)
    {
        printf("%s\n", car[i]);
    }
    return 0;
}
```

Output:

```
Mahindra
Audi
Lamborghini
Hyundai
Tata Motors
```

Conclusion

- An array of strings in C is a one-dimensional array of strings and a two-dimensional array of characters.

- We can declare the array of strings by pointer method (char*) or by using 2d notations.
- The initialization of the array of strings is simple, we can either provide a string wrapped inside double quotes to create an array of pointers, or we can write a nested array of characters.
- When we create an array of strings by using char* notation/pointer approach, all strings are internally interpreted as a constant character array, so we cannot change it later but in the case of an array of strings by 2d notations the string pointers are constant, so we could change their content but not that address they pointing.
- While accessing strings we cannot access invalid or out-of-bound memory. This might either create undefined behaviour or error.