## Welcome, PROGRAMMERS



01.

**What is Character?** 



### What is

### Character?



### Character



The **char** (character) data type is used to **represent single characters**.

The char data type is usually 1 byte in size, and it can hold a

single character from the ASCII character set.

To represent character data type, **%c** is used as the format specifier.

02.

What is ASCII?



### What is

**ASCII?** 



### ASCII



Characters in C are internally represented as ASCII (American Standard Code for Information Interchange) values.

#### For example,

- the character 'A' has an ASCII value of 65,
- the character 'B' has a value of 66, and so on.

### **ASCII** Representation



To represent a character in an **ASCII representation**, use **%d** instead of %c while printing.

char letter = 'a'; printf("**%c**", letter);

Output: a

char letter = 'a'; printf("**%d**", letter);

Output: 97

### **ASCII VALUES**



#### Common ASCII value range:

Characters	ASCII Values
A to Z	65 to 90
a to z	97 to 122
0 to 9	48 to 57
Space / NULL	32

03.

What is String?



### What is

### String?



### String



A **string** is **an array of characters** terminated by a **null character \0**.

Strings in C are represented as character arrays, where each element of the array is a character in the string, and the **null** 

character \0 indicates the end of the string.

Let's see **syntax** of a **String** in detail with some examples...

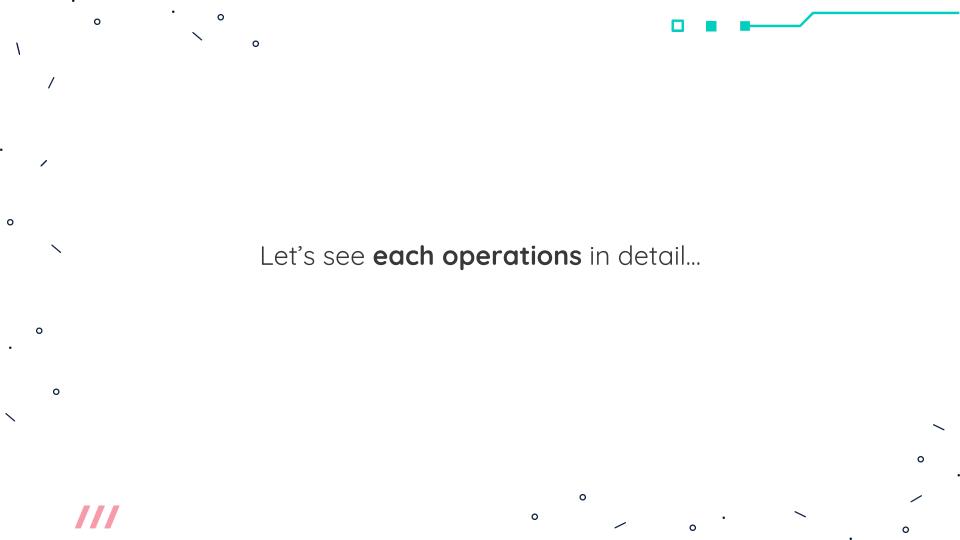
# **Syntax of String** char array\_name[size];

### String Operations



There are many operations can be perform on a string. But, here are the **most common operations** of Array:







	Elements					
char a[5] = {	'h',	'e',	11,	<b>'1'</b> ,	603	};
Index / Position	0	1	2	3	4	

**Predefined String** 



	Elements					
char a[5];						
Index / Position	0	1	2	3	4	

**Empty String** 



char	a[5];				
a[0]	=	'h';			
2[ <b>1</b> ]	_	(0).			

		Elements				
char a[5]	h	е	1	1	0	
Index / Position	0	1	2	3	4	

a[2] = '1'; a[3] = '1'; a[4] = '0';

Index-wise static insertion



	Elements					
char a[5];						
Index / Position	0	1	2	3	4	

**Empty String** 



		Elements				
char a[5];	h	е	1	1	0	
Index / Position	0	1	2	3	4	

scanf("%c", &a[0]); // h
scanf("%c", &a[1]); // e
scanf("%c", &a[2]); // l
scanf("%c", &a[3]); // l
scanf("%c", &a[4]); // o

char a[5];

Index-wise dynamic insertion

## O2 Iteration Operation

#### **Iteration Operation**



char a[5] = {'h', 'e', 'l',
 'l', 'o'};

printf("%c", a[0]); // h
printf("%c", a[1]); // e
printf("%c", a[2]); // l
printf("%c", a[3]); // l
printf("%c", a[4]); // o

	Elements						
char a[5]	h	е	1	1	0		
Index / Position	0	1	2	3	4		

Index-wise static accessing of elements

### **Iteration Operation**



char a[5] = {'h',	'e',	·1 <sup>,</sup>
'l', 'o'};		
int i;		

	Elements						
char a[5]	h	е	1	1	0		
Index / Position	0	1	2	3	4		

101	(1-0) 1 - 1		' /
{			
	printf("%c	,, ,	a[ <b>i</b> ])
}			

for(i=0: i<=4: i++)

Index-wise dynamic accessing of elements



### Modification/Updation Operation

### **Updation Operation**



	Elements					
char a[5] = {	'h',	'e',	11,	<b>'1'</b> ,	603	};
Index / Position	0	1	2	3	4	

**Predefined String** 

### **Updation Operation**

a[3] = p;



	Elements					
char a[5]	h	е	1	р	0	
Index / Position	0	1	2	3	4	

Index-wise static updation

### **Updation Operation**



scanf("%c", &a[1]); // i

	Elements						
char a[5]	h	i	1	р	0		
Index / Position	0	1	2	3	4		

Index-wise dynamic updation



Let's start now...



