

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:

Insertion

1

Iteration

2

Modification /  
Updation

3





Let's see **each operations** in detail...





01

# Insertion Operation



# Insertion Operation

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

Predefined String

# Insertion Operation

	Elements					
<code>char a[5];</code>						
<b>Index / Position</b>	0	1	2	3	4	

Empty String



# Insertion Operation

```
char a[5];
```

```
a[0] = 'h';
```

```
a[1] = 'e';
```

```
a[2] = 'l';
```

```
a[3] = 'l';
```

```
a[4] = 'o';
```

	Elements					
char a[5]	h	e	l	l	o	
Index / Position	0	1	2	3	4	

Index-wise static insertion

# Insertion Operation

	Elements					
<code>char a[5];</code>						
<b>Index / Position</b>	0	1	2	3	4	

Empty String

# Insertion Operation

char a[5];

	Elements					
char a[5];	h	e	l	l	o	
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
```

Index-wise dynamic insertion



02

# 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	e	l	l	o
Index / Position	0	1	2	3	4

Index-wise static accessing of  
elements

# Iteration Operation

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

```
for(i=0; i<=4; i++)  
{  
    printf("%c ", a[i]);  
}
```

	Elements				
char a[5]	h	e	l	l	o
Index / Position	0	1	2	3	4

Index-wise dynamic accessing of  
elements



03

# Modification/Updation Operation



# Updation Operation

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

Predefined String



# Updation Operation

`a[3] = 'p';`

	Elements				
char a[5]	h	e	l	p	o
Index / Position	0	1	2	3	4

Index-wise static updation

# Updation Operation

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

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

Index-wise dynamic updation



# Language

Let's start now...

