

Shaping "skills" for "scaling" higher...!!!

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?





Shaping "skills" for "scaling" higher...!!!



C LANG. 4





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





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





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.

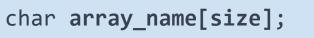


Let's see **syntax** of a **String** in detail with some examples...



Syntax of String







STRING OPERATIONS



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

Insertion Iteration Modification / Updation

1 2 3



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









		Elements				
char a[5] = {	'h',	e,	'1' ,	'1',	°o'	};
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];	

$$a[2] = '1';$$

$$a[3] = 1;$$

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

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]); // 1 scanf("%c", &a[3]); // 1

scanf("%c", &a[4]); // o

Index-wise dynamic insertion



char a[5];



Iteration Operation



Iteration Operation



```
char a[5] = {'h', 'e', 'l',
(1', '0');
printf("%c", a[0]); // h
printf("%c", a[1]); // e
printf("%c", a[2]); // 1
printf("%c", a[3]); // 1
```

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', 'l',
  'l', 'o'};
int i;
```

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

	(- 0) - 1	, – ·	• /
{			
	<pre>printf("%</pre>	c ",	a[i]);
}			

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

Index-wise dynamic accessing of elements





Modification/Updation Operation



Updation Operation



		Elements				
char a[5] = {	ʻh',	e,	'1' ,	'1' ,	°o'	};
Index / Position	0	1	2	3	4	

Predefined String



Updation Operation



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

Index-wise static updation



a[3] =

Updation Operation



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

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

Index-wise dynamic updation







Let's start now...





0