ALX LESSON 0x06 C - More pointers, arrays and strings

C - Programming

TABLE OF CONTENTS

01

OVERVIEW topics

Topics

What are pointers and how to use them

What are arrays and how to use them

What are the differences between pointers and arrays

How to use strings and how to manipulate them

Scope of variables

Slides On Telegram

https://t.me/alx_2023

C Programming

Topics

02

Learning Objectives

Data Types

Data Type	Size (bytes)	Range of Values
`char`	1	-128 to 127 or 0 to 255 (unsigned)
`short`	2	-32,768 to 32,767 or 0 to 65,535 (unsigned)
`int`	4	-2,147,483,648 to 2,147,483,647 or 0 to 4,294,967,295 (unsigned)
`long`	4	-2,147,483,648 to 2,147,483,647 or 0 to 4,294,967,295 (unsigned)
`long long`	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 or 0 to 18,446,744,073,709,551,615 (unsigned)
`float`	4	approximately 1.2E-38 to 3.4E+38
`double`	8	approximately 2.2E-308 to 1.8E+308
`long double`	12 or more	depends on implementation

Data Types

Data Type	Size (bytes)	Range of Values	Format Specifier	Signed/Unsigned
`char`	1	-128 to 127 or 0 to 255 (unsigned)	`%c`	Signed/Unsigned
`short`	2	-32,768 to 32,767 or 0 to 65,535 (unsigned)	`%hd`	Signed
`int`	4	-2,147,483,648 to 2,147,483,647 or 0 to 4,294,967,295 (unsigned)	`%d` or `%i`	Signed
`long`	4	-2,147,483,648 to 2,147,483,647 or 0 to 4,294,967,295 (unsigned)	`%ld`	Signed
`long long`	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 or 0 to 18,446,744,073,709,551,615 (unsigned)	`%lld`	Signed
`float`	4	approximately 1.2E-38 to 3.4E+38	`%f`	Signed
`double`	8	approximately 2.2E-308 to 1.8E+308	`%lf`	Signed
`long double`	12 or more	depends on implementation	`%Lf`	Signed

Data Types (byte = 8 bit)

Data Type	Format Specifier	Minimal Range	Typical Bit Size
unsigned char	%c	0 to 255	8
char	%c	-127 to 127	8
signed char	%c	-127 to 127	8
int	%d, %i	-32,767 to 32,767	16 or 32
unsigned int	%u	0 to 65,535	16 or 32
signed int	%d, %i	Same as int	Same as int 16 or 32
short int	%hd	-32,767 to 32,767	16
unsigned short int	%hu	0 to 65,535	16
signed short int	%hd	Same as short int	16

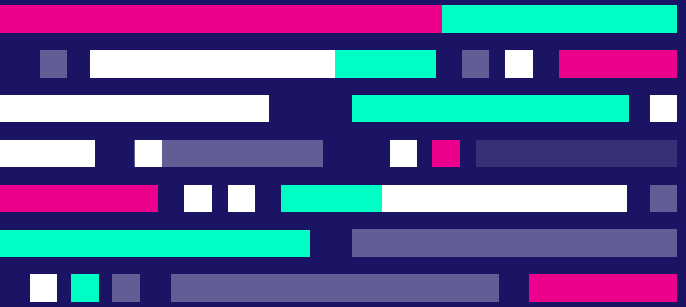
Data Types (byte = 8 bit)

long int	%ld, %li	-2,147,483,647 to 2,147,483,647	32
long long int	%lld, %lli	-(263 – 1) to 263 – 1 (It will be added by the C99 standard)	64
signed long int	%ld, %li	Same as long int	32
unsigned long int	%lu	0 to 4,294,967,295	32
unsigned long long int	%llu	264 – 1 (It will be added by the C99 standard)	64
float	%f	1E-37 to 1E+37 along with six digits of the precisions here	32
double	%lf	1E-37 to 1E+37 along with six digits of the precisions here	64
long double	%Lf	1E-37 to 1E+37 along with six digits of the precisions here	80

Specifiers

SPECIFIER	USED FOR
%c	a single character
%s	a string
%hi	short (signed)
%hu	short (unsigned)
%Lf	long double
%n	prints nothing
%d	a decimal integer (assumes base 10)
%i	a decimal integer (detects the base automatically)
%o	an octal (base 8) integer
%x	a hexadecimal (base 16) integer
%p	an address (or pointer)
%f	a floating point number for floats
%u	int unsigned decimal
%e	a floating point number in scientific notation
%E	a floating point number in scientific notation
%%	the % symbol

C – Strings



Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello.“

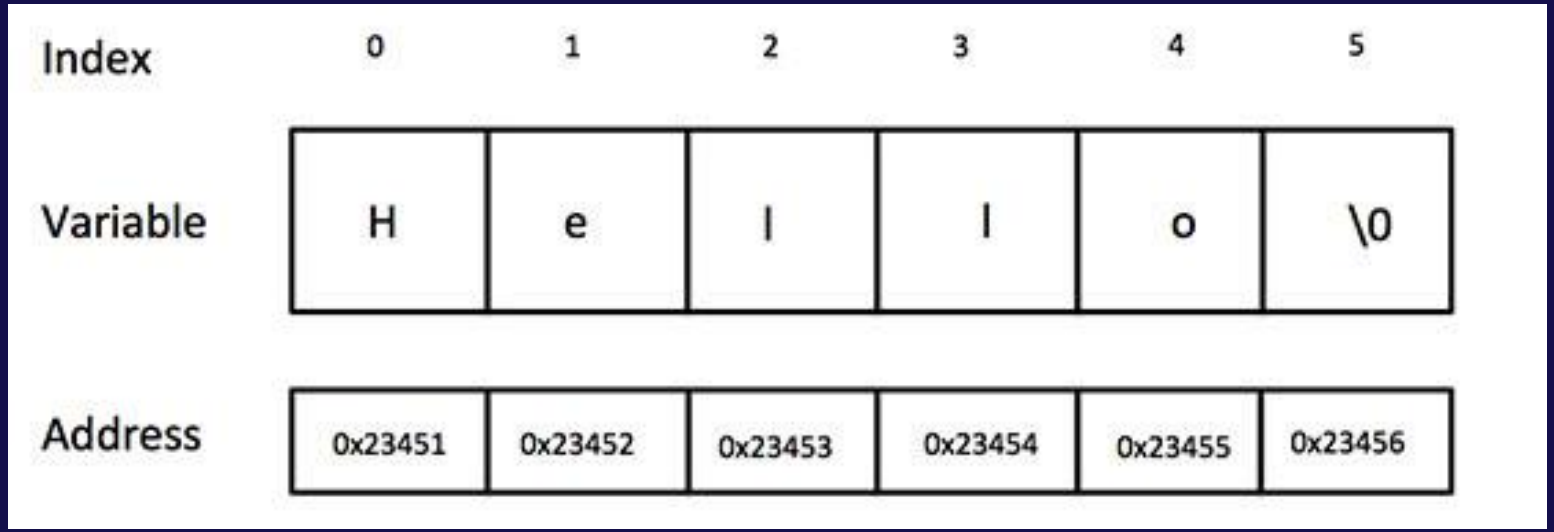
```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0’};
```

If you follow the rule of array initialization then you can write the above statement as follows -

```
char greeting[] = "Hello";
```

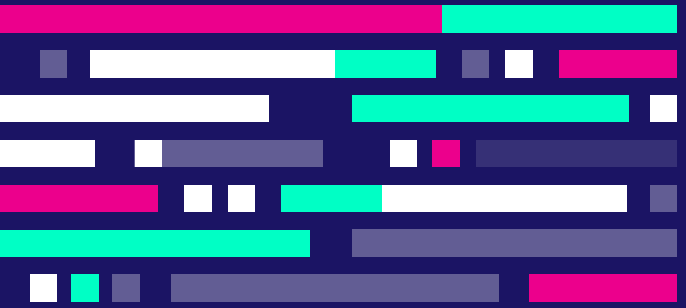
C – Strings

Following is the memory presentation of the above defined string in C/C++ :



Actually, you do not place the null character at the end of a string constant. The C compiler automatically places the '\0' at the end of the string when it initializes the array. Let us try to print the above mentioned string

Declaration of Strings



char str_name[size];

In the above syntax str_name is any name given to the string variable and size is used to define the length of the string, i.e the number of characters strings will store.

Note: There is an extra terminating character which is the Null character ('\0') used to indicate the termination of a string that differs strings from normal character arrays. When a Sequence of characters enclosed in the double quotation marks is encountered by the compiler, a null character '\0' is appended at the end of the string by default.

Initializing a String

4 Ways to Initialize a String in C

1. Assigning a string literal without size: String literals can be assigned without size. Here, the name of the string str acts as a pointer because it is an array.

```
char str[] = "Hello String";
```

2. Assigning a string literal with a predefined size: String literals can be assigned with a predefined size. But we should always account for one extra space which will be assigned to the null character. If we want to store a string of size n then we should always declare a string with a size equal to or greater than n+1.

```
char str[50] = "Hello String";
```

Initializing a String

3. Assigning character by character without size: We can assign character by character without size with the NULL character at the end. The size of the string is determined by the compiler automatically.

```
char str[] = {'H', 'e', 'l', 'l', 'o', ' ', 'S', 't', 'r', 'i', 'n', 'g','\0'};
```

4. Assigning character by character with size: We can also assign a string character by character. But we should remember to set the end character as '\0' which is a null character.

```
char str[13] = {'H', 'e', 'l', 'l', 'o', ' ', 'S', 't', 'r', 'i', 'n', 'g','\0'};
```

C program to illustrate strings

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

int main()
{
    // declare and initialize string
    char str[] = "Hello String";

    // print string
    printf("%s\n", str);

    int length = 0;
    length = strlen(str);

    // displaying the length of string
    printf("Length of string str is %d", length);

    return 0;
}
```

How to Read a String From User?

```c
// read string from user
#include<stdio.h>

int main()
{
    // declaring string
    char str[50];

    // reading string
    scanf("%s",str);

    // print string
    printf("%s",str);

    return 0;
}
```

Passing Strings to Function

```c
// pass string to functions
#include <stdio.h>

void printStr(char str[]) { printf("String is : %s", str); }

int main()
{
    // declare and initialize string
    char str[] = "Hello Strings";

    // print string by passing string
    // to a different function
    printStr(str);

    return 0;
}
```

Strings and Pointers

In Arrays, the variable name points to the address of the first element. Similar to Arrays in C we can create a character pointer to a string that points to the starting address of the string which is the first character of the string. The string can be accessed with the help of pointers as shown in the below example.

Strings and Pointers

```c
#include <stdio.h>

int main()
{

    char str[20] = "Hello World";

    // Pointer variable which stores the starting address of the character array str
    char* ptr = str;

    // While loop will run till the character value is not equal to null character
    while (*ptr != '\0') {
        printf("%c", *ptr);

        // moving pointer to the next character.
        ptr++;
    }

    return 0;
}
```

Most Used Functions in C Strings

strlen(string_name)	Returns the length of string name.
strcpy(s1, s2)	Copies the contents of string s2 to string s1.
strcmp(str1, str2)	Compares the first string with the second string. If strings are the same it returns 0.
strcat(s1, s2)	Concat s1 string with s2 string and the result is stored in the first string.
strstr(s1, s2)	Find the first occurrence of s2 in s1.
strchr(s1, ch)	Returns a pointer to the first occurrence of character ch in string s1.

Most Used Functions in C Strings Ex:

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

int main() {
   char s1[100] = "Hello";
   char s2[100] = "World";
   char s3[100] = "Hello";
   char str1[100] = "abc";
   char str2[100] = "xyz";
   char string_name[100] = "Some text";
   char ch = 'e';

   // Using strlen()
   int len = strlen(string_name);
   printf("Length of string '%s' is %d\n", string_name, len);

   // Using strcpy()
   strcpy(s3, s2);
   printf("After copying s2 to s3, s3 is now '%s'\n", s3);

   // Using strcmp()
   int cmp = strcmp(str1, str2);
   if (cmp == 0) {
      printf("'%s' is equal to '%s'\n", str1, str2);
   } else if (cmp < 0) {
      printf("'%s' is less than '%s'\n", str1, str2);
   } else {
      printf("'%s' is greater than '%s'\n", str1, str2);
   }

   // Using strcat()
   strcat(s1, s2);
   printf("After concatenating s2 to s1, s1 is now '%s'\n", s1);

   // Using strstr()
   char* ptr = strstr(s1, s2);
   printf("Substring '%s' found in '%s' at position %ld\n", s2, s1, ptr - s1);

   // Using strchr()
   char* ptr2 = strchr(s1, ch);
   printf("Character '%c' found in '%s' at position %ld\n", ch, s1, ptr2 - s1);

   return 0;
}
```

What is the ASCII character set

```
cook@pop-os:~$ ascii -d
   0 NUL    16 DLE    32      48 0    64 @    80 P    96 `   112 p
   1 SOH    17 DC1    33 !    49 1    65 A    81 Q    97 a   113 q
   2 STX    18 DC2    34 "    50 2    66 B    82 R    98 b   114 r
   3 ETX    19 DC3    35 #    51 3    67 C    83 S    99 c   115 s
   4 EOT    20 DC4    36 $    52 4    68 D    84 T   100 d   116 t
   5 ENQ    21 NAK    37 %    53 5    69 E    85 U   101 e   117 u
   6 ACK    22 SYN    38 &    54 6    70 F    86 V   102 f   118 v
   7 BEL    23 ETB    39 '    55 7    71 G    87 W   103 g   119 w
   8 BS     24 CAN    40 (    56 8    72 H    88 X   104 h   120 x
   9 HT     25 EM     41 )    57 9    73 I    89 Y   105 i   121 y
  10 LF     26 SUB    42 *    58 :    74 J    90 Z   106 j   122 z
  11 VT     27 ESC    43 +    59 ;    75 K    91 [   107 k   123 {
  12 FF     28 FS     44 ,    60 <    76 L    92 \   108 l   124 |
  13 CR     29 GS     45 -    61 =    77 M    93 ]   109 m   125 }
  14 SO     30 RS     46 .    62 >    78 N    94 ^   110 n   126 ~
  15 SI     31 US     47 /    63 ?    79 O    95 _   111 o   127 DEL
```

Hexadecimal Numbering System

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

04

Hands on lab Practice


Have a Question
Leave a Comment!

Share

To let the others know more

Subscribe

To stay updated with latest videos

Thanks