Declaration of String and Initialization

There are many ways to declare a string:

Method 1:

```
char state [ 6 ] ;   // just a declaration of an array
```

is an array of characters. We initialize the individual cells using single quote characters, like

```
state[ 0 ] = 'U' ;

state[ 1 ] = 't'   ;

state[ 2 ] = 'a'  ;

state[ 3 ] = 'h'  ;

state[ 4 ] = '\0' ;  // IT IS OUR RESPONSIBILITY TO ASSIGN A NULL
```

Note, the last cell should be tagged with NULL character '\0' so we know the end of the string. The NULL character will not be part in the computation of the length of the string. For instance, the length of the string above is 4 (excluding the NULL character) .

Method 2:

In this definition,

```
char states [ 6 ] = "UTAH " ;  // PAY ATTENTION TO THIS.
```

There are several things happening behind the scenes here. First, a constant string is made in memory "UTAH" . Secondly, the array variable "states" is defined and allocated. Thirdly, the array is initialized by copying the constant string "UTAH" . In effect, we are defining a two strings: one constant and one array of characters.

Note here , we defined a string "UTAH" with double quotes, not single quotes. constant strings should be within double quotes.   When you change the contents of the array, you are not changing the constant string, but the contents of the array only. The constant string cannot be changed

METHOD 3

In this method, we use the scanf function. Remember, scanf function requires the address of the variable to be passed along with the conversion specifier. In strings, the conversion specifier is %s referring to strings. We know by declaration of an array, the array name is itself a pointer constant and it is also the address of the first cell.

Luckily, we just have to pass this array name as the address of the array.

For instance,

```
char state [ 6 ]  ;
```

in the above definition, state is the array name and is also the address of the first cell and is also a pointer constant.

Now, our scanf function becomes

```
scanf ( "%s", state ) ;
```

not `scanf ( "%s",  &state ) ; // which is incorrect`

METHOD 4:

In this method, we simply use the strcpy function to initialize the array.   The strcpy function takes two parameters in the form

```
char *     strcpy(char *destinationBuffer,  char *sourceBuffer

);

char state[6] = "UTAH" ;

char myState [6 ] ;


strcpy ( myState,  state ) ;
```

The destination string should be as big as the source or can be longer. The strcpy copies including the NULL character from the source buffer. The function also returns the pointer to the destination array.

METHOD 5:

Method 5:

Much similar to method 1, we can declare and initialize an string as

```
 char states [ ]  = { 'U', 't', 'a', 'h', '\0' } ;
```

Here we declare 5 cell array and initialize the cells using the character including the NULL character.

But if do

```
char states [  6 ]  = { 'U', 't', 'a', 'h' } ;
```

then you can skip the NULL characters because the fifth and the sixth cell will automatically assigned zero, the NULL character.

String Library

Sample code

```
// THE CODE WE WROTE IN THE VIDEO

    char str1[20] = "CSUS STATE" ;

    char ch = 'b' ;


    if ( strchr ( str1, ch ) )

      printf ( "ch is present str1 \n");

    else // NULL  = 0

      printf ( "ch is not present str1 \n");
```

The various functions we would use in programs are:

Video on strchr

https://youtu.be/uYUWf_YYE4U (https://youtu.be/uYUWf_YYE4U)

(https://youtu.be/uYUWf_YYE4U)

How to use strcat and strcpy :

https://youtu.be/OoTvf1PPaQ0 (https://youtu.be/OoTvf1PPaQ0) [(https://youtu.be/OoTvf1PPaQ0)](https://youtu.be/OoTvf1PPaQ0)

How to use strstr:

https://youtu.be/S_uAtSdXnso [(https://youtu.be/S_uAtSdXnso)](https://youtu.be/S_uAtSdXnso)

[(https://youtu.be/S_uAtSdXnso)](https://youtu.be/S_uAtSdXnso)

Sample code to use strstr

The prototype of strstr function is defined as

```
char *   strstr(const char *s1, const char *s2) ;
```

The strstr() function locates the first occurrence of the null-terminated string s2 in the null-terminated string s1. The strcasestr() function is similar to strstr(), but ignores the case of both strings.

RETURN VALUES FROM MAN PAGES: If s2 is an empty string, s1 is returned; if s2 occurs nowhere in s1, NULL is returned; otherwise a pointer to the first character of the first occurrence of s2 is returned.

```
// THE CODE WE WROTE IN THE VIDEO

        char str1[20] = "Sac State";
```

```
    char str2[20] = "ate";


  if ( strcasestr ( str1, str2) ) // THIS IGNORES CASES

     printf ("%s is a substring of %s\n", str2, str1);

  else // NULL

     printf ("%s is not a substring of %s\n", str2, str1);
```

How could we implement strstr :

[strstr implementation (https://www.youtube.com/watch?v=NtYJ49ZsNhY)](https://www.youtube.com/watch?v=NtYJ49ZsNhY)

[(https://www.youtube.com/watch?v=NtYJ49ZsNhY)](https://www.youtube.com/watch?v=NtYJ49ZsNhY)

ctype.h functions

To use these functions, include

#include <ctype.h>

In many programming tasks, you may want to determine if a character is a letter, digit, symbol and so on.

Types of character sets in our ASCII Table are

- alphabetic characters : A-Z, a-z
- alpha numeric character : A-Z, a-z, 0-9
- digits : 0 – 9
- hexadecimal 0-9, A-F, a-f
- lower case a – z, upper case A – Z
- punctuation - ~ @ # $ % ^ & * ( ) - _ + { } | \ < > ? / : ; " ' , .
- space characters
- control characters

int   isalpha (int value) ;

The isalpha() function returns non-zero if the argument value is >=65 and <=90 and >= 97 and <=122 tests false and returns zero otherwise.

int tolower (int value) ;

This converts the upper case letter to lower case letter. if the letter is not upper case, the value is returned unchanged.

int toupper (int value) ;

This converts the lower case letter to upper case letter. if the letter is not lower case, the value is returned unchanged.

int islower (int value) ;

The islower ( ) function returns non-zero if the argument is a lower case letter and returns zero if the argument tests true

int isupper (int value) ;

The isupper ( ) function returns non-zero if the argument is a upper case letter and returns zero if the argument tests true

int isalnum (int value) ;

This function check if the argument is any character in the set A-Za-Z0-9. In a way, we can rewrite this function as

```
if ( isdigit ( ch ) || isalpha ( ch ) )
    return non-zero ;
else
    return zero ;
```

int isprint (int c) ; This functions tests if the character is printable or not. The following are printable characters

<space> ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

int ispunct (int c) ; this function checks if the character is a punctuation character or not . The following are punctuation characters

! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~

int iscntrl (int c) ; This function checks if the argument is a control character. A control character is any ASCII value from 0 to 31 and the ASCII value 127.

To use all these functions, you want to include the header file

<ctype.h>

t y p e s ---------- c h a r a c t e r --------- s e t s (https://www.youtube.com/watch?v=cxABDXqC63M)

(https://www.youtube.com/watch?v=cxABDXqC63M)