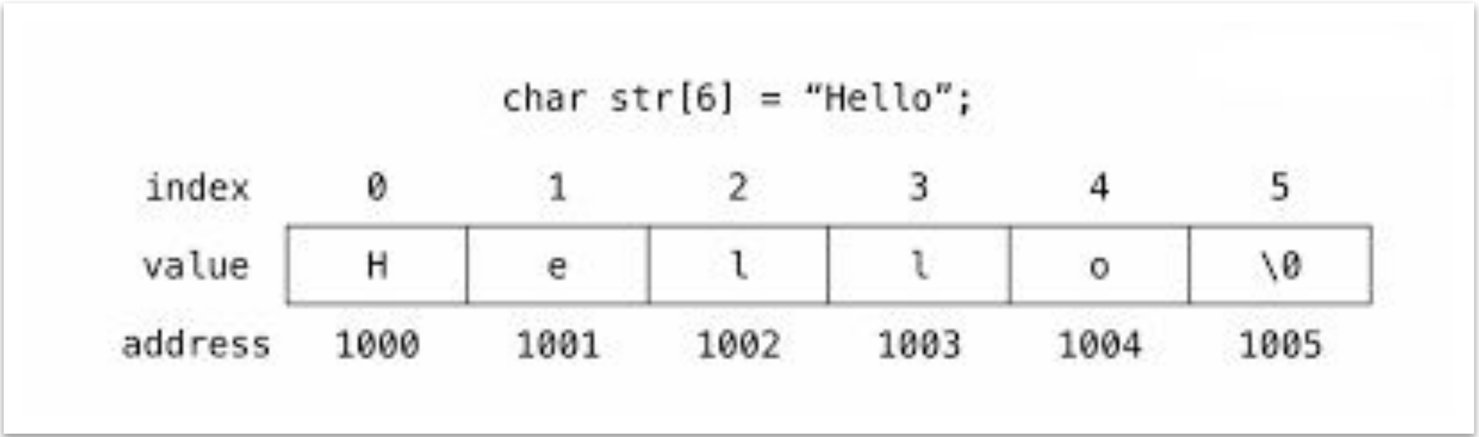
Chapter 8 - Characters and Strings

8.1 Introduction

- Strings
  - A **string** in **C** is an array of **characters**.
  - The length of a **string** is determined by a terminating null **character**: '\0' .
  - So, a **string** with the contents, say, “hello" has six **characters**: ‘h' , ‘e' , ‘l' , ‘l' , ‘o' and the terminating null ('\0‘) **character**.
  - The terminating null **character** has the value zero.



index	0	1	2	3	4	5
value	H	e	l	l	o	\0
address	1000	1001	1002	1003	1004	1005

8.1 Introduction

- Series of characters treated as a single unit
  - Can include letters, digits and special characters (*, /, $)
- String literal (string constant) - written in double quotes
  - "Hello“
- String a pointer to first character
- Value of string is the address of first character

8.2 Fundamentals of Strings

- String definitions
  - Define as a character array or a variable of type char *

    ```
    char color[] = "blue";
    char *colorPtr = "blue";
    ```
  - Remember that strings represented as character arrays end with '\0'
    - color has 5 elements
- Inputting strings
  - Use scanf

    ```
    scanf("%s", word);
    ```
    - Copies input into word[]
    - Do not need & (because a string is a pointer)
  - Remember to leave room in the array for '\0'

```
#include "stdafx.h"
void main()
{
	char colour[] = "blue";
	printf("colour=%s", colour);
}
```

```
#include "stdafx.h"
void main()
{
	char *colour = "blue";
	printf("colour=%s", colour);
}
```

```
#include "stdafx.h"
void main()
{
	char *colour = "blue";
	puts(colour);
}
```

```
#include "stdafx.h"
void main()
{
    char colour[] = "blue";
    int i = 0;
    while (colour[i] != '\0')
    {
        printf("%c", colour[i]);
        i++;
    }
}
```

```
#include "stdafx.h"
void main()
{
    char *color = "blue";
    while (*color != '\0')
    {
        printf("%c", *color); //or putchar(*color);
        color++;
    }
}
```

8.3 Character Handling Library

- Character handling library
  - Includes functions to perform useful tests and manipulations of character data
- The following slide contains a table of all the functions in <ctype.h>

8.3 Character Handling Library

Prototype	Description
`int isdigit( int c );`	Returns `true` if `c` is a digit and `false` otherwise.
`int isalpha( int c );`	Returns `true` if `c` is a letter and `false` otherwise.
`int isalnum( int c );`	Returns `true` if `c` is a digit or a letter and `false` otherwise.
`int isxdigit( int c );`	Returns `true` if `c` is a hexadecimal digit character and `false` otherwise.
`int islower( int c );`	Returns `true` if `c` is a lowercase letter and `false` otherwise.
`int isupper( int c );`	Returns `true` if `c` is an uppercase letter; `false` otherwise.
`int tolower( int c );`	If `c` is an uppercase letter, `tolower` returns `c` as a lowercase letter. Otherwise, `tolower` returns the argument unchanged.
`int toupper( int c );`	If `c` is a lowercase letter, `toupper` returns `c` as an uppercase letter. Otherwise, `toupper` returns the argument unchanged.
`int isspace( int c );`	Returns `true` if `c` is a white-space character—newline (`'\n'`), space (`' '`), form feed (`'\f'`), carriage return (`'\r'`), horizontal tab (`'\t'`), or vertical tab (`'\v'`)—and `false` otherwise
`int iscntrl( int c );`	Returns `true` if `c` is a control character and `false` otherwise.
`int ispunct( int c );`	Returns `true` if `c` is a printing character other than a space, a digit, or a letter and `false` otherwise.
`int isprint( int c );`	Returns `true` value if `c` is a printing character including space (`' '`) and `false` otherwise.
`int isgraph( int c );`	Returns `true` if `c` is a printing character other than space (`' '`) and `false` otherwise.

```
#include "stdafx.h"
#include "ctype.h"

void main()
{
      char c;

      printf("Enter a character: ");
      scanf_s("%c", &c, 1); // or c = getchar();

      if (isdigit(c) )
      printf("%c is a digit.\n", c);
      else
      printf("%c is not a digit.\n", c);
}
```

```
Enter a character: 4
4 is a digit.
```

Having to put a 1 as the size of a single character.

Characters

- Characters are simple alphabets like a, b, c, d...., A, B, C, D,....., any single digit number like 0, 1, 2,....and special characters like $, %, +, -.... etc., are also treated as characters and to assign them in a character type variable,

- you simply need to put them inside **single quotes**.

- For example, the following statement defines a character type variable **ch** and we assign a value 'a' to it −

char ch = 'a';

```c
#include "stdafx.h"
void main()
{
    char c;
    printf("Input No.1\n");
    scanf_s("%c", &c,1);
    printf("c = %c\n", c);

    printf("Input No.2\n");
    scanf_s("%c", &c,1);
    printf("c = %c\n", c);

    printf("Input No.3\n");
    scanf_s("%c", &c,1);
    printf("c = %c\n", c);
}
```

```
Input No.1
s
c = s
Input No.2
c =

Input No.3
a
c = a
```

As you see, the input No.2 was skipped. As a result, first scanf will read the s. Second scanf will read the enter! That's why, the second printf of the value of c leaves just a newline after "c=". Then the third scanf waits for a key press. You input a and then you hit enter. a is been assigned to variable c and enter remains in the stdin buffer, ready to be read by the next scanf. If we had a fourth scanf, then it would read the enter.

So, just change scanf_s("%c", &c,1); to scanf_s(" %c", &c,1);
and you will be just fine. You can see in the code below:

```
#include "stdafx.h"
void main()
{
    char c;
    printf("Input No.1\n");
    scanf_s("%c", &c,1);
    printf("c = %c\n", c);

    printf("Input No.2\n");
    scanf_s(" %c", &c,1);
    printf("c = %c\n", c);

    printf("Input No.3\n");
    scanf_s(" %c", &c,1);
    printf("c = %c\n", c);
}
```

```
Input No.1
s
c = s
Input No.2
a
c = a
Input No.3
m
c = m
```

8.4 String Functions

- Used to manipulate character and string data

Function	Function description
`getchar`	Inputs the next character from the standard input and returns it as an integer.
`gets`	Inputs characters from the standard input into the array `s` until a newline or end-of-file character is encountered. A terminating null character is appended to the array.
`putchar`	Prints the character stored in `c`.
`puts`	Prints the string `s` followed by a newline character.
`sprintf_s`	Equivalent to `printf`, except the output is stored in the array `s` instead of printing it on the screen.
`sscanf_s`	Equivalent to `scanf`, except the input is read from the array `s` instead of reading it from the keyboard.

```
#include "stdafx.h"
#include "string.h"
void main()
{
     char c;        /* variable to hold character input by user */
     char sentence[80]; /* create char array */
     int i = 0;

     /* prompt user to enter line of text */
     puts("Enter a line of text:");

     /* use getchar to read each character */
     while ((c = getchar()) != '\n') {
     sentence[i++] = c;
     }

     sentence[i] = '\0'; /* terminate string */

     /* use puts to display sentence */
     puts("\nThe line entered was:");
     puts(sentence);
}
```

```
Enter a line of text:
This is a test.

The line entered was:
This is a test.
```

```c
#include "stdafx.h"

void main()
{
    char s[80]; /* create char array */
    int x;          /* x value to be input */
    double y;       /* y value to be input */

    printf("Enter an integer and a double:\n");
    scanf_s("%d%lf", &x, &y);

    sprintf_s(s, "integer:%6d\ndouble:%8.2f", x, y);

    printf("%s\n%s\n","The formatted output stored in
    array s is:", s);
}
```

```
Enter an integer and a double:
298 87.375
The formatted output stored in array s is:
integer:   298
double:   87.38
```

```
#include "stdafx.h"

void main()
{
    char s[] = "31298 87.375"; /* initialize array s */
    int x;    /* x value to be input */
    double y; /* y value to be input */

    sscanf_s(s, "%d%lf", &x, &y);

    printf("%s\n%s%6d\n%s%8.3f\n","The values stored in character
    array s are:", "integer:", x, "double:", y);
}
```

```
The values stored in character array s are:
integer: 31298
double:   87.375
```

8.5 String Manipulation Functions of the String Handling Library

- String handling library has functions to
  - Manipulate string data

Function prototype	Function description
`strcpy_s(s1, s2 )`	Copies string `s2` into array `s1`. The value of `s1` is returned.
`strncpy_s( s1, s2, n)`	Copies at most `n` characters of string `s2` into array `s1`. The value of `s1` is returned.
`strcat_s( s1, s2 )`	Appends string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.
`strncat_s(s1, s2, n )`	Appends at most `n` characters of string `s2` to array `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.

```
#include "stdafx.h"
#include "string.h"
void main()
{
    char x[] = "Happy Birthday to You";
    char y[25];
    char z[15];

    printf("%s%s\n", "The string in array x is: ", x);
    /* copy contents of x into y */
    strcpy_s(y, x);
    printf("The string in array y is: %s\n", y);

    /* copy first 14 characters of x into z. Does not copy null
    character */
    strncpy_s(z, x, 14);

    z[14] = '\0'; /* terminate string in z */
    printf("The string in array z is: %s\n", z);
}
```

```
The string in array x is: Happy Birthday to You
The string in array y is: Happy Birthday to You
The string in array z is: Happy Birthday
```

```
#include "stdafx.h"
#include "string.h"
void main()
{
     char s1[20] = "Happy ";
     char s2[] = "New Year ";
     char s3[40] = "";         /* initialize char array s3 to empty */

     printf("s1 = %s\ns2 = %s\n", s1, s2);

     /* concatenate s2 to s1 */
     strcat_s(s1, s2);
     printf("strcat( s1, s2 ) = %s\n",s1 );

     /* concatenate first 6 characters of s1 to s3. Place '\0' after last character */
     strncat_s(s3, s1, 6);
     printf("strncat( s3, s1, 6 ) = %s\n",s3 );

     /* concatenate s1 to s3 */
     strcat_s(s3, s1);
     printf("strcat( s3, s1 ) = %s\n",s3 );
}
```

```
s1 = Happy
s2 = New Year
strcat( s1, s2 ) = Happy New Year
strncat( s3, s1, 6 ) = Happy
strcat( s3, s1 ) = Happy Happy New Year
```

8.6 Comparison Functions of the String Handling Library

- Comparing strings
  - Computer compares numeric ASCII codes of characters in string

strcmp(s1, s2);

- Compares string s1 to s2
- Returns a negative number if s1 < s2, zero if s1 == s2 or a positive number if s1 > s2

strncmp(s1, s2, n);

- Compares up to n characters of string s1 to s2
- Returns values as above

```c
#include "stdafx.h"
#include "string.h"
void main()
{
    char s1[] = "Happy New Year";
    char s2[] = "Happy New Year";
    char s3[] = "Happy Holidays";

    printf("%s%s\n%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n\n",
    "s1 = ", s1, "s2 = ", s2, "s3 = ", s3,
    "strcmp(s1, s2) = ", strcmp(s1, s2),
    "strcmp(s1, s3) = ", strcmp(s1, s3),
    "strcmp(s3, s1) = ", strcmp(s3, s1));

    printf("%s%2d\n%s%2d\n%s%2d\n",
    "strncmp(s1, s3, 6) = ", strncmp(s1, s3, 6),
    "strncmp(s1, s3, 7) = ", strncmp(s1, s3, 7),
    "strncmp(s3, s1, 7) = ", strncmp(s3, s1, 7));
}
```

```
s1 = Happy New Year
s2 = Happy New Year
s3 = Happy Holidays

strcmp(s1, s2) =  0
strcmp(s1, s3) =  1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) =  0
strncmp(s1, s3, 7) =  1
strncmp(s3, s1, 7) = -1
```

8.7 Other Functions

- strlen(s);
  - Returns the number of characters (before NULL) in string s
- _strrev(s) function
  - strrev( ) function reverses a given string s.

```c
#include "stdafx.h"
#include "string.h"
void main()
{
    char s[30] = "Hello";

    printf("String before reverse : %s\n", s);

    printf("String after reverse  : %s\n", _strrev(s));

}
```

```
String before reverse : Hello
String after reverse  : olleH
```

```
#include "stdafx.h"
void main()
{
    char input[64];
    int size, c;

    printf("enter a common earth phrase:");
    gets(input);

    puts("\nHere is how we say that on bacward
    planet:");
    size = strlen(input);
    for (c = size - 1; c >= 0; c--)
    putchar(input[c]);
    printf("\n");
}
```

```
enter a common earth phrase:Just Do It

Here is how we say that on bacward planet:
tI oD tsuJ
```

8.8 Exercises

This program reads the word character by character and then prints out the string

```
#include "stdafx.h"
void main()
{
    char str[20] = "hello";
    int i = 0;
    while (str[i] != '\0')
    {
    putchar(str[i]);
    i++;
    }
    printf("\n");
}
```

This program reads characters until a newline, stores them in an array and terminates the string with a NULL character. It then prints out the string.

```
#include "stdafx.h"
void main()
{
     char str[20], ch;
     int i = 0;
     printf("enter some characters:\n");
     ch = getchar();
     while (ch != '\n')
     {
          str[i] = ch;   //*(str+i)=ch;
          i++;
          ch = getchar();
     }
     str[i] = '\0'; // *(str+i)=NULL; or *(str+i)=0;

     printf("\nthe string is:\n");
     i = 0;
     while (str[i] != '\0')
     {
          putchar(str[i]); // putchar(*(str+i));
          i++;
     }
     printf("\n");
}
```

Passing array to a function

```
#include "stdafx.h"
void func(char *p)
{
    int i = 0;
    while (*(p + i) != '\0')
    {
        putchar(*(p + i));
        i++;
    }
    printf("\n");
    }
void main()
{
    char str[25];
    printf("enter a message:\n");
    gets(str);
    func(str);
}
```

```
#include "stdafx.h"
void main()
{
    char *article[] = { "the", "a", "one", "some", "any" };
    for (int i = 0; i<5; i++)

        printf("%s\n", *(article + i));

}
```

```
the
a
one
some
any
```

```
#include "stdafx.h"
void main()
{
    char *name[] = { "usman","reza","metin" };
    char *task[] = { "task1", "task2", "task3" };
    char sentence[50] = "";
    srand(time(NULL));
    for (int i = 0; i <= 2; i++)
    {
        strcat(sentence, name[i]);
        strcat(sentence, "==>");
        strcat(sentence, task[rand() % 3]);
        printf("%s\n\n", sentence);
        sentence[0] = NULL;
    }
}
```

```
usman==>task2

reza==>task1

metin==>task3
```