

# STRINGS

**Week 8**

# Strings

- A **string** is any sequence of characters *enclosed* in double quotes  
"Salam Shabab".
- There is no separate data type for strings as **char**, **integer**, **float** or **double**.
- Instead, a string is represented in **C** as an array of type **char**.

# Strings

- We can declare and initialize a string variable using any of the following:

```
char str1[20] = {'S','a','l','a','m','
```

```
','S','h','a','b','a','b', '\0'}; //as other arrays
```

```
char str2[20] = "Salam Shabab"; //with size
```

```
char str3[]= "Salam Shabab"; //without size
```

- The string is terminated by the `\0` character:

# Strings

- For example, the declaration:

```
char str[20] = "Salam Shabab";
```

is actually represented in the memory as shown below:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
S	a	l	a	m		S	h	a	b	a	b	\0	?	?	?	?	?	?	?

# Input / Output of strings

- We can use %s to read and print a string

```
printf ("Name is %s \n", Name);
```

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

# Input / Output of strings

- We can use %s to read and print a string

```
printf ("Name is %s \n", Name);
```

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

*Where is* **&**



# Input / Output with gets and puts

- A problem with scanf when reading a string is that it stops scanning the moment it encounters a white space.
- Thus, it cannot scan a string such as: “Ali Zaki Street” in one variable.
- An alternative to scanf is the function `gets` that takes a string variable as argument.

```
char school[SIZE];  
gets(school);
```

# Input / Output with gets and puts

- A problem with scanf when reading a string is that it stops scanning the moment it encounters a white space.
- Thus, it cannot scan a string such as: “Ali Zaki Street” in one variable.
- An alternative to scanf is the function `gets` that takes a string variable as argument.

```
char school[SIZE];  
gets(school);
```

No &





# Input / Output with gets and puts

- The `gets` function continues to scan for characters until it encounters the new line character – until the user types the Enter key.
- The function `puts` can be used to print a string.

```
puts(school);
```

- **Note:-** What is the difference between:

`gets(s)` and `scanf("%s",s)` when `s` exceeds the limit ?

# String.h

- To use some operations or functions on strings, we need to include file string.h

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

- So, we can use functions
  - strcpy(s1, s2)
  - strlen(s)
  - strcmp(s1, s2)
  - strcat (s1, s2)

# String Functions

`strcpy(s1, s2)`

- we can not use operator “ = ” with string
- **strcpy** is copying s2 into s1 (s1 ← s2)

# String Functions

`strlen(S)`

- Return the length of string S
- `strcpy(S, "industrial robot");`
- `strlen(S)` will return 16

# String Functions

strcmp(s1, s2)

- It is for comparison of s1 and s2
- It return 0 if s1=s2
- Return -ve if *s1 is less than s2*
- Return +ve if *s1 is greater than s2*

# String Functions

`strcat(s1, s2)`

- It appends/concatenates `s2` to the end of `s1`

Example: `s1 = "Lion"`, `s2 = "King"`

`strcat(s1,s2) → s1="LionKing"`

# Example (1)

```
#include <stdio.h>
#include <string.h>
int main()
{   int k;
    char src  [30];
    char dest [10]=" Tutorial";
    strcpy(src, "This is a");
    strcat(src, dest);
    k=strlen(src);
    printf("Final string : %s, with length = \n", src,k);
    return(0); }
```

# Example (2)

// Compare two strings and determine which is greater

```
gets(s);
```

```
gets(d);
```

```
k=strcmp(s,d);
```

```
if (k>0)
```

```
    printf("\n First is less than second \n");
```

```
else if (k==0)
```

```
    printf("\n First equals second \n");
```

```
else if (k<0)
```

```
    printf("\n First is greater than second \n");
```



# Passing string to function

- When a string is used as a function parameter it is:

Call by Value

,

Call by reference

- Can the return value of a function be a string (array of character)?
- |

# Array of Strings

- To represent arrays of strings we need **2-dimensional arrays** of characters.
- The first-dimension represents the number of strings in the array and the second-dimension represents the strings.
- The following are statements to declare an array to store up to 30 names, each of maximum length, 25 characters.

```
#define NUM_NAMES 30
```

```
#define NAME_LEN 25
```

```
...
```

```
char names[NUM_NAMES][NAME_LEN];
```

# Array of Strings

- We can also initialize an array of strings at declaration in the following manner:
- `char month[12][10] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};`

# Example

```
// Print array of strings
#include <stdio.h>
int main()
{   int i;
    char Names[3][20]={"Akram Ali" , "Mido Zaki",
                      "Rami Mohsen" , "Said Abdo"};
    for (i=0; i<3; i++)
        printf("%s\n",Names[i]);

    return(0);}
```