# Strings

# Character array initialization

char s[] = {‘I’,’ ’,’a’,’m’,’ ’,’d’,’o’,’n’,’e’,’\0’}; or

char s[] = “I am done”; //String constant

- Both the statements are equivalent.
- Null character ‘\0’ is automatically added to the end.

# Strings

Strings are:

- Character array
- Terminated by ‘\0’ (null) character. This is required as certain pre-defined functions need this to work.
- The \0 is not part of the string. Thus, it will not be counted in the length of string.
- char s[] = “Hello”;

- But this is not allowed:

char s[] = “Hello”;//Fine

s[] = “Hello”; //Worng

# Printing Strings

- Strings are printed using %s option of printf.

```
printf("%s", "I am done"); or
for(i=0; i<size; i++)
        printf("%c", arr[i]);
```

```
char str[] = "I am done";
```

Note: In this case, NULL character is added in the end.

# Guess the output

```
char str[] = “I am done”;
str[5] = ‘\0’;
printf(“%s”, str); //I am
scanf(“%s”, str);

for(int i=0; i<11; i++)
        putchar(str[i]);        //I am one
```

# Reading a String(scanf)

- Placeholder: %s
- Argument: Name of character array (Why? Recall Call-by-Reference)
- No & sign before character array as it holds the base address.
- With %s, scanf skips the whitespaces
  - starts with first non-whitespace character.
  - Copies into the array one-by-one.
  - Continue till a whitespace character is reached.
- Places the null character at the end of the string.

# Example

```
#include<stdio.h>
void main() {
        char str1[20], str2[20];
        scanf(“%s”, str1);
        scanf(“%s”, str2);
        printf(“%s + %s\n”, str1, str2);
}
```

Input: IISC Bangalore

Output: IISC + Bangalore

Input: I am done

Output: I + am

# Read_Line function

- Write a function while reads a string till \n.

```
int read_line(char str[]) {
        char c;
        int i=0;
        c = getchar();
        while(c != '\n' && c != EOF) {
                str[i] = c;
                i++;
                c = getchar();
        }
        str[i] = '\0';
        return i;
}
```

The problem with the above code is lack of synchronization between size of string and input.

# String Copy

```
char s1[] = “Welcome to CSVTU”;
char s2[];
s2 = s1; //Error: Array initializer must be a list or a string.
```

- To copy an array/string, we need to do element-wise copying.

```
int a = 10, b;
b = a;
```

# str_copy function

- Arguments: Two strings: destination (dest) and source (src).
- Copy contents of src into dest.
- We assume that dest is declared with size at least as large as src;
- Note: the use of ‘\0’ for loop termination.

```
void str_copy(char dest[], char src[]) {
        int i;
        for(i=0; src[i] != ‘\0’; i++) {
                dest[i] = src[i];
        }
        dest[i] = ‘\0’;
}
```

# Compare Two Strings

- Lexicographical Ordering

Order of words in a dictionary.

- Alphabetical sequence (Dictionary sequence)

- “mat” is smaller than “matter”.
- “cap” is smaller than “cat”.

# str_compare function

```
int str_compare(char str1[], char str2[]) {
        int i=0;
        while(srt1[i] == str2[i]) {
                if(str1[i] == '\0' || str2[i] == '\0')
                        break;
                i++;
        }
        if(str1[i] == str2[i])
                return 0;
        else if(str1[i] < str2[i])
                return -1;
        else
                return 1;
}
```

Returns:
0, if strings are equal
-1, if str1 is smaller
1 if str2 is smaller

# String Functions (predefined)

- Return length of a string.
- Concatenates one string with another.
- Search for a substring in a given string.
- Reverse a string.
- Find first/last/k-th occurrence of a character in a string.
- Case sensitive/ insensitive versions

HELLO != hello case sensitive

HELLO == hello case insensitive

# string.h header file

- Header file with predefined helper functions for strings.
- strlen(s): returns length of string s without \0.
- strcpy(d,s): copies s into d.
- strcat(d, s): appends s at the end of d (\0 is moved at the end of result).
- strcmp(s1, s2): return an integer less than or equal to, or greater than zero if s1 is found, respectively, to be less than, to match, or be greater than s2.

Example:

```
char str1[]="Hello", str2[]="Helpo";
int i = strcmp(str1, str2);
printf("%d", i); //Output: -4 which is 'l' – 'p'
```

# Contd.

- strncpy(d, s, n)
- strncat (d, s, n)
- strncmp(d, s, n)
- Restrict the function to “n” characters atmost (n is integer).
- First two functions truncate the string to first n characters/
- Third function truncates d and s both to first n characters.

char str1[] = “Hello”, str2[] = “Helpo”;

printf(“%d”, strncmp(str1, str2, 3)); //Output: 0

# Contd.

- strcasecmp and strncasecmp

Case insensitive comparison.  hello == HELLO == Hello == hEllo

```
char str1[] = “HELLO”, str2[] = “hello”, str3[] = ”Helpo”;
int i = strcmp(str1, str3);
int j = strcasecmp(str1, str2);
printf(“%d %d”, i, j);
```

# Utility Functions

- strupr(string): converts lower case to upper case. hello => HELLO
- strlwr(string): converts upper case to lower case. HELLO => hello
- strstr(Source, key): function to search key into Source. Returns a pointer to the first occurrence.

```
 #include <stdio.h>
#include <string.h>
void main() {
        char S[] = "abababababa“, s[] = "ba";
        printf("%s", strstr(S, s));
}
```

Output: bababababa

# Exercise

- Write a C function to change the case of text from lower to upper.

  str_upper(s) = upper case

- Write a C function to concatenate two strings.

  str_concat(s1, s2)  s1 + s2

  hello hi

  hellohi