Department of
Computer Science and Engineering

Title: String processing

Structured Programming Lab
CSE 104

Green University of Bangladesh

1 Learning Objective

- To attain knowledge on string, string declaration, string initialization, string library function
- To use string functions to process C-strings
- To implement programs using String

2 Strings in C

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.
For example:

```
char c[] = "c string";
```

When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.

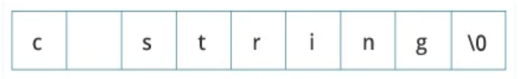


Figure 1: Memory Diagram

2.1 Declare a string

Declaring a string is as simple as declaring a one-dimensional array.

2.1.1 Syntax for declaring a string

```
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.
Please keep in mind that there is an extra terminating character which is the Null character \0 used to indicate the termination of string which differs strings from normal character arrays.

2.2 Initializing a String

A string can be initialized in different ways. For example:

```
char c[] = "abcd";

char c[50] = "abcd";

char c[] = {'a', 'b', 'c', 'd', '\0'};

char c[5] = {'a', 'b', 'c', 'd', '\0'};
```

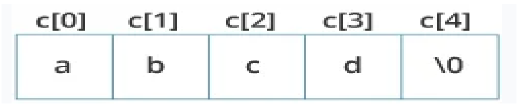


Figure 2: Memory Representation

Let us now look at a sample program to get a clear understanding of declaring and initializing a string in C and also how to print a string.
Code:

```c
// C program to illustrate strings

#include<stdio.h>

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

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

    return 0;
}
```

We can see in the above program that strings can be printed using normal printf statements just like we print any other variable. Unlike arrays, we do not need to print a string, character by character. The C language does not provide an inbuilt data type for strings but it has an access specifier "%s" which can be used to directly print and read strings.

2.3 Read String from the user

2.3.1 Example 1: Using scanf

You can use the scanf() function to read a string. The scanf() function reads the sequence of characters until it encounters whitespace (space, newline, tab, etc.).
Code:

```c
// C program to read strings

#include<stdio.h>

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

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

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

    return 0;
}
```

We know that the '&' sign is used to provide the address of the variable to the scanf() function to store the value read in memory. As str[] is a character array so using str without braces '['and']' will give the base address of this string. That's why we have not used '&' in this case as we are already providing the base address of the string to scanf.

2.3.2 Example 2: Using fgets() and puts()

You can use the fgets() function to read a line of string. And, you can use puts() to display the string.
Code:

```c
// C program to read strings
#include <stdio.h>
int main()
{
    char name[30];
    printf("Enter name: ");
    fgets(name, sizeof(name), stdin);  // read string
    printf("Name: ");
    puts(name);    // display string
    return 0;
}
```

The sizeof(name) results to 30. Hence, we can take a maximum of 30 characters as input which is the size of the name string. To print the string, we have used puts(name);.
The gets() function can also be to take input from the user. However, it is removed from the C standard. It's because gets() allows you to input any length of characters. Hence, there might be a buffer overflow.

2.4 Passing strings to function

Strings can be passed to a function in a similar way as arrays. Below is a sample program to do this:
Code:

```c
// C program to illustrate how to
// pass string to functions
#include<stdio.h>

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

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

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

    return 0;
}
```

3 Strings and Pointers

Strings can also be declared using pointer.In the following example we are creating a string str using char character array of size 6.

```c
char str[6] = "Hello";
```

The above string can be represented in memory as follows.

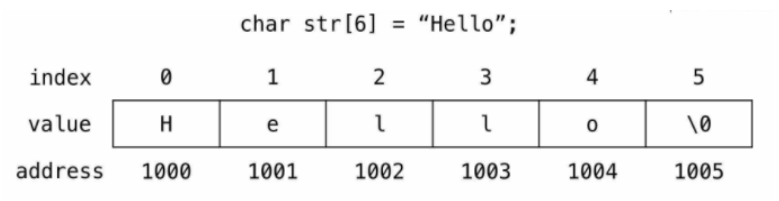


Figure 3: Memory Diagram

Each character in the string str takes 1 byte of memory space.

3.1 Creating a pointer for the string

The variable name of the string str holds the address of the first element of the array i.e., it points at the starting memory address.

So, we can create a character pointer ptr and store the address of the string str variable in it. This way, ptr will point at the string str. In the following code we are assigning the address of the string str to the pointer ptr.

```c
char *ptr = str;
```

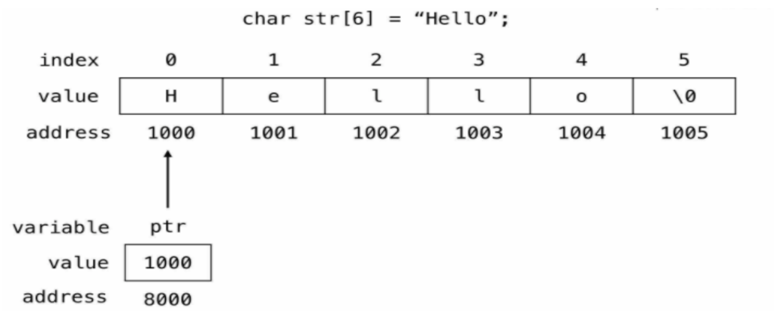


Figure 4: Memory Diagram

The pointer variable ptr is allocated memory address 8000 and it holds the address of the string variable str i.e., 1000.

3.2 Accessing string via pointer

To access and print the elements of the string we can use a loop and check for the '\0' null character. In the following example we are using while loop to print the characters of the string variable str.

Code:

```c
#include <stdio.h>

int main() {

  // string variable
  char str[6] = "Hello";

  // pointer variable
  char *ptr = str;

  // print the string
  while(*ptr != '\0') {
    printf("%c", *ptr);

    // move the ptr pointer to the next memory location
    ptr++;
  }

  return 0;
}
```

Another example:

Code:

```c
#include <stdio.h>

int main() {
  char name[] = "Harry Potter";

  printf("%c", *name);      // Output: H
  printf("%c", *(name+1));    // Output: a
  printf("%c", *(name+7));    // Output: o

  char *namePtr;

  namePtr = name;
  printf("%c", *namePtr);      // Output: H
  printf("%c", *(namePtr+1));    // Output: a
  printf("%c", *(namePtr+7));    // Output: o
}
```

Table 1: Strings Library Functions

Function	Work of Function	Syntax
strlen()	computes string's length	int strlen(char *string);
strcpy()	copies a string to another	char *strcpy (char *dest, char *src);
strcat()	concatenates(joins) two strings	char *strcat(char *dest, const char *src);
strcmp()	compares two strings	int strcmp(char *string1, char *string2);
strlwr()	converts string to lowercase	char *strlwr(char *str);
strupr()	converts string to uppercase	char *strupr(char *str);
strrev()	Reverses the string	Char * strrev(char *string);
strchr()	Find first occurrence of character c in string	char *strchr(char *string, int c);
strstr()	Find first occurrence of string string1 in string2	char *strstr(char *string2, char *string1);
strtok()	Parse the string s into tokens using delim as delimiter	char *strtok(char *s, const char *delim) ;

4 Commonly Used String Functions

You need to often manipulate strings according to the need of a problem. Most, if not all, of the time string manipulation can be done manually but, this makes programming complex and large.
To solve this,C supports a large number of string handling functions in the standard library "string.h". Few commonly used string handling functions are discussed in **Table1**.

Example 1:
Code:

```c
// c program to demonstrate
// example of strlen() function.

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

int main()
{
   char ch[]={'H', 'e', 'l', 'l', 'o', '\0'};

   printf("Length of string is: %d", strlen(ch));

 return 0;
}
```

Output:

```
Length of string is: 5
```

Example 2:
Code:

```c
// c program to demonstrate
// example of strcmp() function.

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

int main () {
   char str1[20];
   char str2[20];
   int result;

   //Assigning the value to the string str1
   strcpy(str1, "hello");

   //Assigning the value to the string str2
   strcpy(str2, "hEllo");

   result = strcmp(str1, str2);

```

```
    if(result > 0) {
        printf("ASCII value of first unmatched character of str1 is greater than str2");
    } else if(result < 0) {
        printf("ASCII value of first unmatched character of str1 is less than str2");
    } else {
        printf("Both the strings str1 and str2 are equal");
    }

    return 0;
}
```

Output:

```
ASCII value of first unmatched character of str1 is greater than str2
```

This strcmp() function returns the following values based on the comparison result:

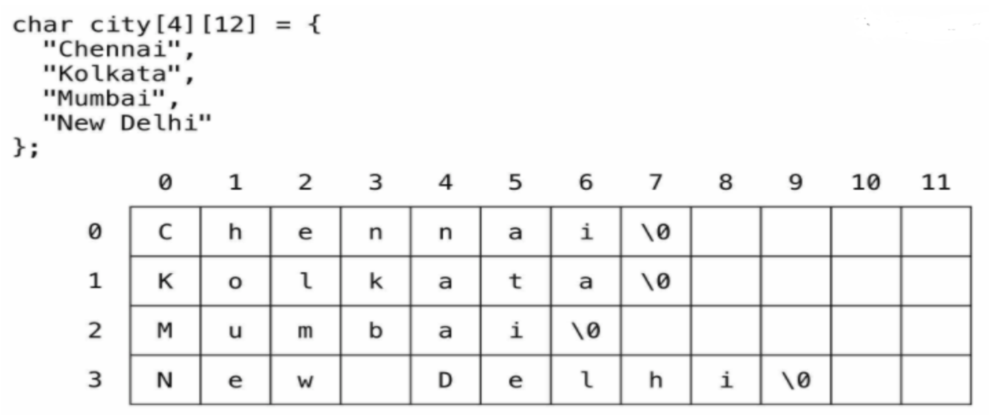
- 0 if both the strings are equal
- >0 if the ASCII value of first unmatched character of string str1 is greater than the character in string str2
- <0 if the ASCII value of first unmatched character of string str1 is less than the character in string str2

5 Array of strings

We can create a two dimensional array and save multiple strings in it.
For example, in the given code we are storing 4 cities name in a string array city.

```
char city[4][12] = {
  "Chennai",
  "Kolkata",
  "Mumbai",
  "New Delhi"
};
```

We can represent the city array as follows.



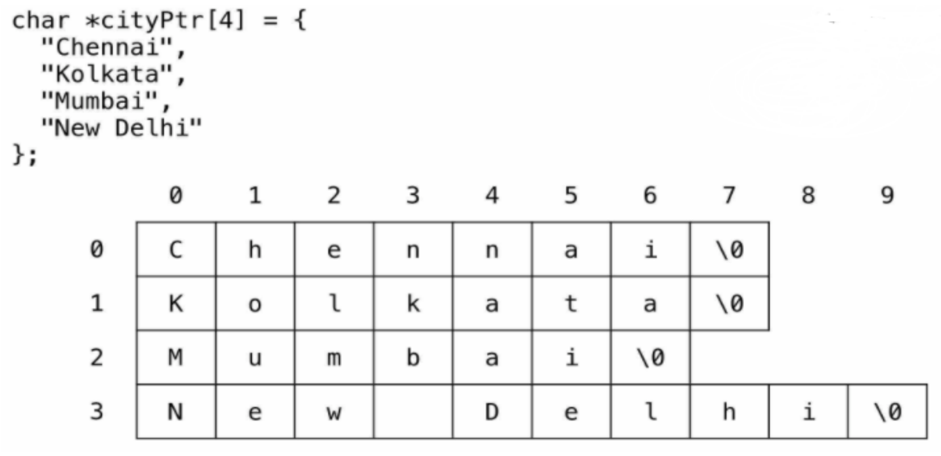
```
char city[4][12] = {
  "Chennai",
  "Kolkata",
  "Mumbai",
  "New Delhi"
};
```

	0	1	2	3	4	5	6	7	8	9	10	11
0	C	h	e	n	n	a	i	\0				
1	K	o	l	k	a	t	a	\0				
2	M	u	m	b	a	i	\0					
3	N	e	w		D	e	l	h	i	\0		

Figure 5: Array Representation

In the above code we are creating an array of character pointer cityPtr of size 4 to store the name of the four cities.

We can represent the array of pointers as follows.



```c
char *cityPtr[4] = {
  "Chennai",
  "Kolkata",
  "Mumbai",
  "New Delhi"
};
```

	0	1	2	3	4	5	6	7	8	9
0	C	h	e	n	n	a	i	\0		
1	K	o	l	k	a	t	a	\0		
2	M	u	m	b	a	i	\0			
3	N	e	w		D	e	l	h	i	\0

Figure 6: Array Representation

6 Lab Task (Please implement yourself and show the output to the instructor)

1. Write a C Program to Copy a String.
2. Write a C Program to Find the Length of a String.
3. Write a C Program to Find the Frequency of Characters in a String.
4. Write a C Program to Reverse a Sentence by Recursion.

7 Lab Exercise (Submit as a report)

1. Write a program in C to find the length of a string without using library function.
2. Write a program in C to count total number of alphabets, digits and special characters in a string.
3. Write a program in C to extract a substring from a given string.
4. Write a program in C to replace the spaces of a string with a specific character.

8 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.