C - STRING

Dr. Sheak Rashed Haider Noori
Professor & Associate Head
Department of Computer Science

C - STRING

- strings are arrays of chars. String literals are words surrounded by double quotation marks.

"This is a static string"

- The string in C programming language is actually a one-dimensional array of characters which is terminated by a **null** character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a **null**.
- A string can be declared as a **character array** or with a **string pointer**.
- The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

Or

```
char greeting[] = "Hello";
```

Or

```
char *greeting = "Hello" ;
```

C - STRING

- Following is the memory presentation of above defined string in C/C++:

Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

- It's important to remember that there will be an extra character on the end on a string, literally a '\0' character, just like there is always a period at the end of a sentence. Since this string terminator is unprintable, it is not counted as a letter, but it still takes up a space. Technically, in a fifty char array you could only hold 49 letters and one null character at the end to terminate the string.
- Actually, you do not place the *null* character at the end of a string constant. The C compiler automatically places the '\0' at the end of the string when it initializes the array.

C - STRING

- Let us try to print above mentioned string:

```
#include <stdio.h>

int main ()
{
   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

   printf("Greeting message: %s\n", greeting );

   return 0;
}
```

When the above code is compiled and executed, it produces result something as follows:

```
Greeting message: Hello
```

- Note: %s is used to print a string.

STRING POINTER

- String pointers are declared as a pointer to a char.
- When there is a value assigned to the string pointer the NULL is put at the end automatically.
- Take a look at this example:

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

void main()
    {
        char *ptr_mystring;

        ptr_mystring = "HELLO";

        printf("%s\n", ptr_mystring);

    }
```

STRING POINTER

- It is not possible to read, with scanf(), a string with a string pointer. You have to use a character array and a pointer. See this example:

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

void main()
    {
            char my_array[10];
            char *ptr_section2;

            printf("Type hello and press enter\n");
            scanf("%s", my_array);
            ptr_section2 = my_array;
            printf("%s\n", ptr_section2);
    }
```

READING A LINE OF TEXT

gets() and puts() are two string functions to take string input from user and display string respectively

```
int main(){
    char name[30];
    printf("Enter name: ");
    gets(name);       //Function to read string from user.
    printf("Name: ");
    puts(name);      //Function to display string.
    return 0;
}
```

STRING RELATED OPERATIONS

- Find the Frequency of Characters in a String
- Find the Number of Vowels, Consonants, Digits and White space in a String
- Reverse a String by Passing it to Function
- Find the Length of a String
- Concatenate Two Strings
- Copy a String
- Remove all Characters in a String except alphabet
- Sort a string in alphabetic order
- Sort Elements in Lexicographical Order (Dictionary Order)
- Change Decimal to Hexadecimal Number
- Convert Binary Number to Decimal

FIND THE FREQUENCY OF CHARACTERS

```c
#include <stdio.h>
int main(){
    char c[1000],ch;
    int i,count=0;
    printf("Enter a string: ");
    gets(c);
    printf("Enter a characeter to find frequency: ");
    scanf("%c",&ch);
    for(i=0;c[i]!='\0';++i)
    {
        if(ch==c[i])
            ++count;
    }
    printf("Frequency of %c = %d", ch, count);
    return 0;
}
```

```
Enter a string: This website is awesome.
Enter a frequency to find frequency: e
Frequency of e = 4
```

C PROGRAM TO FIND FREQUENCY OF CHARACTERS IN A STRING

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

int main()
{
   char string[100];
   int c = 0, count[26] = {0};

   printf("Enter a string\n");
   gets(string);

   while ( string[c] != '\0' )
   {
      /* Considering characters from 'a' to 'z' only */

      if ( string[c] >= 'a' && string[c] <= 'z' )
         count[string[c]-'a']++;

      c++;
   }

   for ( c = 0 ; c < 26 ; c++ )
   {
      if( count[c] != 0 )
         printf("%c occurs %d times in the entered
string.\n",c+'a',count[c]);
   }

   return 0;
}
```

This program computes frequency of characters in a string i.e. which character is present how many times in a string.

For example in the string "code" each of the character 'c', 'o', 'd', and 'e' has occurred one time.

Only *lower case alphabets* are considered, other characters (uppercase and special characters) are ignored. You can easily modify this program to handle uppercase and special symbols.

FIND NUMBER OF VOWELS, CONSONANTS, DIGITS AND WHITE SPACE CHARACTER

```
#include<stdio.h>

int main(){
    char line[150];
    int i,v,c,ch,d,s,o;
    o=v=c=ch=d=s=0;
    printf("Enter a line of string:\n");
    gets(line);
    for(i=0;line[i]!='\0';++i)
    {
        if(line[i]=='a' || line[i]=='e' || line[i]=='i' || line[i]=='o' || line[i]=='u' ||
line[i]=='A' || line[i]=='E' || line[i]=='I' || line[i]=='O' || line[i]=='U')
            ++v;
        else if((line[i]>='a'&& line[i]<='z') || (line[i]>='A'&& line[i]<='Z'))
            ++c;
        else if(line[i]>='0'&&c<='9')
            ++d;
        else if (line[i]==' ')
            ++s;
    }
    printf("Vowels: %d",v);
    printf("\nConsonants: %d",c);
    printf("\nDigits: %d",d);
    printf("\nWhite spaces: %d",s);
    return 0;
}
```

Output

```
Enter a line of string:
This program is easy 2 understand
Vowels: 9
Consonants: 18
Digits: 1
White spaces: 5
```

CALCULATED LENGTH OF A STRING WITHOUT USING STRLEN() FUNCTION

You can use standard library function strlen( ) to find the length of a string but, this program computes the length of a string manually without using strlen( ) funtion.

```
#include <stdio.h>
int main()
{
    char s[1000],i;
    printf("Enter a string: ");
    scanf("%s",s);
    for(i=0; s[i]!='\0'; ++i);
    printf("Length of string: %d",i);
    return 0;
}
```

```
Enter a string: Programiz
Length of string: 9
```

REVERSE STRING

```c
#include<stdio.h>
#include<string.h>
void Reverse(char str[]);
int main(){
     char str[100];
     printf("Enter a string to reverse: ");
     gets(str);
     Reverse(str);
     printf("Reversed string: ");
     puts(str);
 return 0;
}
void Reverse(char str[]){
     int i,j;
     char temp[100];
     for(i=strlen(str)-1,j=0; i+1!=0; --i,++j)
     {
          temp[j]=str[i];
     }
     temp[j]='\0';
     strcpy(str,temp);
}
```

To solve this problem, two standard library functions strlen() and strcpy() are used to calculate length and to copy string respectively.

```
Enter a string to reverse: zimargorp
 Reversed string: programiz
```

CONCATENATE TWO STRINGS MANUALLY

You can concatenate two strings using standard library function strcat( ) , this program concatenates two strings manually without using strcat( ) function.

```
#include <stdio.h>
int main()
{
    char s1[100], s2[100], i, j;
    printf("Enter first string: ");
    scanf("%s",s1);
    printf("Enter second string: ");
    scanf("%s",s2);
    for(i=0; s1[i]!='\0'; ++i);   /* i contains length of string s1. */
    for(j=0; s2[j]!='\0'; ++j, ++i)
    {
        s1[i]=s2[j];
    }
    s1[i]='\0';
    printf("After concatenation: %s",s1);
    return 0;
}
```

```
Enter first string: lol
Enter second string: :)
After concatenation: lol:)
```

COPY STRING MANUALLY

```c
#include <stdio.h>
int main()
{
    char s1[100], s2[100], i;
    printf("Enter string s1: ");
    scanf("%s",s1);
    for(i=0; s1[i]!='\0'; ++i)
    {
        s2[i]=s1[i];
    }
    s2[i]='\0';
    printf("String s2: %s",s2);
    return 0;
}
```

```
Enter String s1: programiz
String s2: programiz
```

You can use the strcpy( ) function to copy the content of one string to another but, this program copies the content of one string to another manually without using strcpy( ) function.

REMOVE CHARACTERS IN STRING EXCEPT ALPHABETS

```c
#include<stdio.h>

int main(){
    char line[150];
    int i,j;
    printf("Enter a string: ");
    gets(line);
    for(i=0; line[i]!='\0'; ++i)
    {
        while (!((line[i]>='a'&&line[i]<='z') ||
(line[i]>='A'&&line[i]<='Z' || line[i]=='\0')))
        {
            for(j=i;line[j]!='\0';++j)
            {
                line[j]=line[j+1];
            }
            line[j]='\0';
        }
    }
    printf("Output String: ");
    puts(line);
    return 0;
}
```

```
Enter a string: p2'r"o@gram84iz./
Output String: programiz
```

This program takes a string from user and *for* loop executed until all characters of string is checked. If any character inside a string is not a alphabet, all characters after it including null character is shifted by 1 position backwards.

SORT A STRING IN ALPHABETIC ORDER

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void sort_string(char *s)
{
   int c, d = 0, length;
   char *pointer, *result, ch;
   length = strlen(s);
   result = (char*)malloc(length+1);
   pointer = s;
   for ( ch = 'a' ; ch <= 'z' ; ch++ )
   {
      for ( c = 0 ; c < length ; c++ )
      {
         if ( *pointer == ch )
         {
            *(result+d) = *pointer;
            d++;
         }
         pointer++;
      }
      pointer = s;
   }
   *(result+d) = '\0';
   strcpy(s, result);
   free(result);
}

void main()
{
   char string[100];
   printf("Enter some text\n");
   gets(string);
   sort_string(string);
   printf("%s\n", string);
}
```

C program to sort a string in alphabetic order: For example if user will enter a string "programming" then output will be "aggimmnoprr" or output string will contain characters in alphabetical order.

```
Enter some text
game
aegm
```

SORT ELEMENTS IN LEXICOGRAPHICAL ORDER (DICTIONARY ORDER)

```c
#include<stdio.h>
#include <string.h>
int main(){
     int i,j;
     char str[10][50],temp[50];
     printf("Enter 10 words:\n");
     for(i=0;i<10;++i)
          gets(str[i]);
     for(i=0;i<9;++i)
         for(j=i+1;j<10 ;++j){
             if(strcmp(str[i],str[j])>0)
             {
                strcpy(temp,str[i]);
                strcpy(str[i],str[j]);
                strcpy(str[j],temp);
             }
     }
     printf("In lexicographical order: \n");
     for(i=0;i<10;++i){
         puts(str[i]);
     }
return 0;
}
```

This program takes 10 words from user and sorts elements in lexicographical order. To perform this task, two dimensional string is used.

```
Enter 10 words:
fortran
java
perl
python
php
javascript
c
cpp
ruby
csharp

In lexicographical order:
c
cpp
csharp
fortran
java
javascript
perl
php
python
ruby
```

C LIBRARY FUNCTIONS

- C supports a wide range of functions that manipulate null-terminated strings:

S.no	String functions	Description
1	**strcat ( )**	Concatenates str2 at the end of str1.
2	strncat ( )	appends a portion of string to another
3	strcpy ( )	Copies str2 into str1
4	strncpy ( )	copies given number of characters of one string to another
5	strlen ( )	gives the length of str1.
6	strcmp ( )	Returns 0 if str1 is same as str2. Returns <0 if strl < str2. Returns >0 if str1 > str2.
7	strcmpi ( )	Same as strcmp() function. But, this function negotiates case. "A" and "a" are treated as same.
8	strchr ( )	Returns pointer to first occurrence of char in str1.
9	strrchr ( )	last occurrence of given character in a string is found
10	strstr ( )	Returns pointer to first occurrence of str2 in str1.
11	strrstr ( )	Returns pointer to last occurrence of str2 in str1.
12	strdup ( )	duplicates the string
13	strlwr ( )	converts string to lowercase
14	strupr ( )	converts string to uppercase
15	strrev ( )	reverses the given string
16	strset ( )	sets all character in a string to given character
17	strnset ( )	It sets the portion of characters in a string to given character
18	strtok ( )	tokenizing given string using delimiter

Following example makes use of few of the above-mentioned functions:

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

int main ()
{
   char str1[12] = "Hello";
   char str2[12] = "World";
   char str3[12];
   int  len ;

   /* copy str1 into str3 */
   strcpy(str3, str1);
   printf("strcpy( str3, str1) :  %s\n", str3 );

   /* concatenates str1 and str2 */
   strcat( str1, str2);
   printf("strcat( str1, str2):   %s\n", str1 );

   /* total lenghth of str1 after concatenation */
   len = strlen(str1);
   printf("strlen(str1) :  %d\n", len );

   return 0;
}
```

When the above code is compiled and executed, it produces result something as follows:

```
strcpy( str3, str1) :  Hello
strcat( str1, str2):   HelloWorld
strlen(str1) :  10
```

STRCAT( ) FUNCTION

- strcat( ) function concatenates two given strings. It concatenates source string at the end of destination string.
- Syntax for strcat( ) function is given below.
  - char * strcat (char * destination, const char * source);
- Example :
- strcat (str2, str1); - str1 is concatenated at the end of str2.
  strcat (str1, str2); - str2 is concatenated at the end of str1.
- As you know, each string in C is ended up with null character ('\0').
- In strcat( ) operation, null character of destination string is overwritten by source string's first character and null character is added at the end of new destination string which is created after strcat( ) operation.

EXAMPLE PROGRAM FOR STRCAT( )

- In this program, two strings “is fun” and “C tutorial” are concatenated using strcat( ) function and result is displayed as “C tutorial is fun”.

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

int main( )
{
   char source[ ] = " is fun" ;
   char target[ ]= " C tutorial" ;

   printf( "\nSource string = %s", source ) ;
   printf( "\nTarget string = %s", target ) ;

   strcat( target, source ) ;

   printf( "\nTarget string after strcat( ) = %s", target );
}
```

Output:

Source string = is fun
Target string = C tutorial
Target string after strcat( ) = C tutorial is fun