UNIT – 6 Part II
Strings

C Programming (CSC115)
BSc CSIT First Semester (TU)

Prepared by:
Dabbal Singh Mahara
ASCOL (2025)

2 STRINGS

- Strings are array of characters i.e. they are characters arranged one after another in memory. Thus, a character array is called string.
- Each character within the string is stored within one element of the array successively.
- A string is always terminated by a null character (i.e. slash zero \0).

Declaring String Variables

- A string variable is declared as an array of characters.
- Syntax:

  char string_name[size];

  The *size* determines the number of characters in the *string_name*.
- E.g. char name[20];

  char city[15];
- When the compiler assigns a character string to a character array, it automatically supplies a *null character* (‘\0’) at the end of the string. Thus, the *size* should be equal to the maximum number of characters in the string plus one.

Initializing String Variables

- Strings are initialized in either of the following two forms:

```
char name[4]={'R','A','M', '\0'};
char name[]={'R','A','M', '\0'};
char city[9]={'N','E','W', ' ','Y','O','R','K','\0'};
```

OR

```
char name[4]="RAM";
char name[]="RAM";
char city[9]="NEW YORK";
```

R	A	M	\0
name[0]	name[1]	name[2]	name[3]

- When we initialize a character array by listing its elements, the null terminator or the size of the array must be provided explicitly.

```
//String Initialization Example
#include <stdio.h>
#include <conio.h>
int main()
{
char city[9]={'N','E','W',' ','Y','O','R','K','\0'};
int i=0;
while(city[i]!='\0')
   {
   printf("%c\t ",city[i]);
   i++;
   }
getch();
return 0;
}
```

Reading Strings from Terminal

- The input function *scanf* can be used with %s format specification to read in a string of characters.
- E.g.

```
char name[20];
scanf("%s", name);
```

- No ampersand(&) is required before variable name.
- Problem:

  "scanf() terminates its input on the first white space it encounters"

```
#include <stdio.h>
#include <conio.h>
int main()
{
char name[20];
printf("\nEnter your name:");
scanf("%s", name);
printf("\nYour name is %s", name);
getch();
return 0;
}
```

Reading Strings from Terminal…

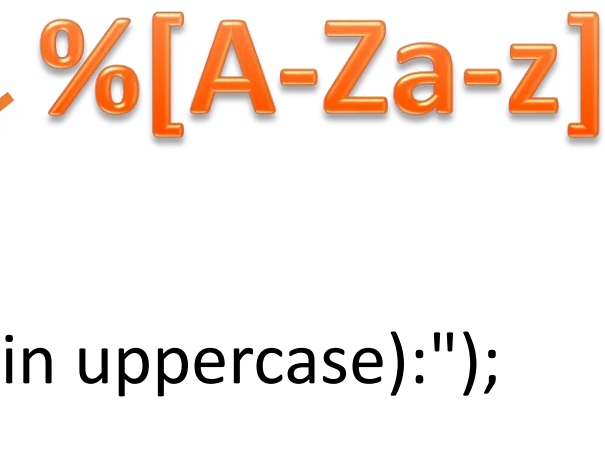
- Some versions of *scanf()* support the following conversion specification for strings:

  %[characters]

  %[^characters]

- The specification *%[characters]* means that only the characters specified within the brackets are allowed in the input of string. If the input string contains any other characters, the reading of string will be terminated at the first encounter of such a character.
- The specification *%[^characters]* means that the characters specified after the caret(^) are not allowed in the string and reading will be terminated.

```
#include <stdio.h>
#include <conio.h>
int main()
{
char name[20];
printf("\nEnter your name (in uppercase):");
scanf("%[A-Z]",name);
printf("\nYour name is %s",name);
getch();
return 0;
}
```



```
/*This program can read strings with blank spaces*/
#include <stdio.h>
#include <conio.h>
int main()
{
char name[30];
printf("\nEnter your full name:");
scanf("%[^\n]", name);
printf("\nYour name is %s", name);
getch();
return 0;
}
```

//Another way using getchar()

```
#include <stdio.h>
#include <conio.h>
int main()
{
char name[30], character;
int c =0;
printf("\nEnter your full name (Press Enter at the end):\n");
do {
   character=getchar();
   name[c]=character;
   c++;
   }
while(character!='\n');
c=c-1;
name[c]='\0';
printf("\nYour name is %s", name);
getch();
return 0;
}
```

Writing Strings to Screen

- The *printf()* function with %s format is used to print strings to the screen.
- The format %s can be used to display an array of characters that is terminated by the null character.
- E.g.

```
char name[19]="……….";
printf("%s", name);
```

Writing Strings to Screen…

- The format specification for outputting strings is of the form,

  % w.p s

  where *w* specifies the field width for display and *p* instructs that only the first p characters of the string are to be displayed.
- By default, the display is right-justified.

Writing Strings to Screen…

- When the field width is less than the length of the string, the entire string is printed.
- The integer value on the right side of the decimal point specifies the number of characters to be printed.
- When the number of characters to be printed is specified as zero, nothing is printed.
- The minus sign in the specification causes the string to be printed left-justified.
- The specification %.ns prints the first n characters of the string (left-justified).

```
#include <stdio.h>
#include <conio.h>
int main()
{
char name[19]=" Jeevan Kumar Thapa";
printf("%19s\n", name);
printf("%10s\n", name);
printf("%19.11s\n", name);
printf("%19.0s\n", name);
printf("%-19.11s\n", name);
printf("%.3s\n", name);
printf("%s\n", name);
getch();
return 0;
}
```

Problem

- Write a program using *for* loop to print the following output:

```
C
CP
CPr
CPro
CProg
…
…
CProgramming
```

```
#include<stdio.h>
#include<conio.h>
int main()
{
char c[13]="CProgramming";
int i, j;
printf("\n\n\n");
printf("-------------\n");
for(i=0;i<=11;i++)
{
 j=i+1;
 printf("%-12.*s\n", j, c);
}
printf("-------------\n");
getch();
return 0;
}
```

Another way: gets() and puts()

- The gets() function is used to read a string of text, containing whitespaces, until a newline character is encountered.
- Syntax: *gets(variable_name);*
- It takes a string from user and stores in a string variable *variable_name*.
- The puts() function is used to display the string onto the screen.
- Syntax: *puts(variable_name);*

```
#include<stdio.h>
#include<conio.h>
Int main()
{
char text[100];
printf("\nWrite whatever:\t");
gets(text);
printf("\nYou have typed:\t");
puts(text);
getch();
return 0;
}
```

Counting length of the string

```
#include <stdio.h>
#include <conio.h>
int main()
{
char input_string[50];
int i=0, length=0;
printf("\nEnter your text:\t");
gets(input_string);
while(input_string[i]!='\0')
    {
            length++;
            i++;
    }
printf("\nThe length of your text is: %d character(s)", length);
getch();
}
```

Another way: strlen()

- The function *strlen()* returns an integer which denotes the length of the string passed into the function.
- The length of the string is defined as the number of characters present in the string, excluding the null character.
- Syntax:

```
integer_variable=strlen(input_string);
```

- Use header file <string.h>.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
   char input_string[50];
   int length;
   printf("\nEnter your text:\t");
   gets(input_string);
   length=strlen(input_string);
   printf("\nThe length of your text is: %d character(s)", length);
   getch();
   return 0;
}
```

```
/* Program to read your name from keyboard and output a list of ASCII codes that
   represent your name */
#include <stdio.h>
#include <conio.h>
int main()
{
   char name[30];
   int len,i;
   printf("Enter your name:\t");
   gets(name);
   len=strlen(name);
   printf("\nThe ASCII values of characters in the name %s are:\n", name);
   for(i=0; i<len; i++)
          printf("%c = %d\n", name[i], name[i]);
   getch();
   return 0;
}
```

/*Program to read a string from keyboard (until enter key) and count the number of vowels, consonants, spaces, semicolons and commas in that string*/

```
#include <string.h>
#include<stdio.h>
int main(){
char s[80];
int len, i, space=0, vowel=0, consonant=0, semicolon=0, comma=0;
printf("Enter string:\t");
gets(s);
len=strlen(s);
for(i=0;i<len;i++)     {
   if(s[i]=='a'||s[i]=='e'||s[i]=='i'||s[i]=='o'||s[i]=='u')
            vowel++;
   else if(s[i]==';')
            semicolon++;
   else if(s[i]==',')
            comma++;
   else if(s[i]==' ')
            space++;
   else
            consonant++;
   }
printf("\nThe number of vowel(s):%d", vowel);
printf("\nThe number of semicolon(s):%d", semicolon);
printf("\nThe number of comma(s):%d", comma);
printf("\nThe number of space(s):%d", space);
printf("\nThe number of consonant(s):%d", consonant);
getch();
}
```

Copying one string to another

```
#include <stdio.h>
#include <conio.h>
int main() {
char copy[50], paste[50];
int i;
printf("\nEnter your name (to copy):\t");
gets(copy);
for(i=0;copy[i]!='\0';i++)
   {
            paste[i]=copy[i];
   }
paste[i]='\0';
printf("\nThe name is (pasted as):\t");
puts(paste);
getch();
}
```

Another way: strcpy()

- The *strcpy()* function copies one string to another.
- It accepts two strings as parameters and copies the second string character by character into the first string including the null character of the second string.
- The source string may be a string constant like “Surkhet”.
- Syntax:

```
strcpy(destination_string, source_string);
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
char copy[50], paste[50];
int i;
printf("\nEnter your name (to copy):\t");
gets(copy);
strcpy(paste, copy);
printf("\nThe name is (pasted as):\t");
puts(paste);
getch();
return 0;
}
```

Combining Strings Together

- Also called String Concatenation.
- Just as we cannot assign one string to another directly, we cannot join two strings together by the simple arithmetic addition. So, the statements such as

  string3=string1+string2;

  string1=string2+“hey”;

  are completely invalid.
- Here, the characters from string1 and string2 should be copied into string3 one after another. Also the size of string3 should be large enough to hold the total characters.

```
#include <string.h>
#include<stdio.h>
main()
{
int i, j, k;
char first_name[10]=“Kumar" ;
char middle_name[10]=“Bikram";
char last_name[10]=“Thapa";
char name[30];
for(i=0;first_name[i]!='\0';i++)
   name[i]=first_name[i];
name[i]=' ';

for(j=0;middle_name[j]!='\0';j++)
            name[i+j+1]=middle_name[j];
   name[i+j+1]=' ';
for(k=0;last_name[k]!='\0';k++)
            name[i+j+k+2]=last_name[k];
   name[i+j+k+2]='\0';
printf("%s", name);
getch();
}
```

Another way: strcat()

- The *strcat()* function concatenates two strings i.e. it appends one string at the end of another.
- It accepts two strings as parameters and stores the contents of the second string at the end of the first string.
- Syntax:

```
strcat(first_string, second_string);
```

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
char first_name[30]=“Arjun” ;
char middle_name[]=“Singh";
char last_name[]=" Thapa";
strcat(first_name,middle_name);
puts(first_name);
strcat(first_name,last_name);
puts(first_name);
getch();
return 0;
}
```

Comparison of Two Strings

- C does not permit the comparison of two strings directly; i.e. statements such as

  ```
  if(name1==name2)
  if(name==“abc”)
  ```

  are not permitted.
- The comparison of two strings has to be done character by character.
- The comparison is done until there is a mismatch or one of the strings terminates into a null character, whichever occurs first.

```
#include <stdio.h>
#include <conio.h>
int main()
{
char str1[30], str2[40];
int i=0;
printf("Enter first string:\t");
gets(str1);
printf("\nEnter second string:\t");
gets(str2);
while(str1[i]==str2[i] && str1[i]!='\0' &&str2[i]!='\0')
   i++;
if(str1[i]=='\0' && str2[i]=='\0')
            printf("\nStrings are equal.");
else
            printf("\nStrings are unequal");
getch();
return 0;
}
```

Another way: strcmp()

- The *strcmp()* function compares two strings to find out whether they are same or different.
- It accepts two strings as parameters and returns an integer 0 if the strings are equal.
- If the two strings are unequal, then it returns an integer that has the numeric difference (ASCII value difference) between the first non-matching characters in the strings.

```
#include <string.h>
#include<stdio.h>
int main()
{
char str1[30],str2[40];
int diff;
printf("Enter first string:\t");
gets(str1);
printf("\nEnter second string:\t");
gets(str2);
diff=strcmp(str1, str2);
if(diff>0)
   printf("\n%s is greater than %s by ASCII value difference %d", str1, str2, diff);
else if(diff<0)
   printf("\n%s is smaller than %s by ASCII value difference %d", str1, str2, diff);
else
   printf("\n%s is same as %s", str1, str2);
getch();
return 0;
}
```

Reversing a String

```
#include <stdio.h>
#include <conio.h>
int main()
{
char string[25], reverse_string[25];
int length, i, j;
printf("\nInput string to be reversed:");
gets(string);
length=strlen(string);
for(j=0,i=length-1;j<length;j++,i--)
        reverse_string[j]=string[i];
reverse_string[j]='\0';
puts(reverse_string);
getch();
return 0;
}
```

Another Way: strrev()

- The function *strrev()* is used to reverse all characters in a string except the null character at the end of string.
- E.g. The reverse of string “BIM” is “MIB”.
- Syntax:

```
strrev(string);
```

- Here, the characters in the string “string” are reversed and the reversed string is stored in “string”.

```
#include <stdio.h>
#include <conio.h>
int main()
{
char string[25];
printf("\nInput string to be reversed:");
gets(string);
strrev(string);
printf("\nThe reversed string is: %s", string);
getch();
return 0;
}
```

```
/*Program to read a string and check for palindrome*/
#include <stdio.h>
#include <conio.h>
int main()
{
char string[50];
int len, i, palindrome=1;
printf("Enter string:\t");
gets(string);
len=strlen(string);
for(i=0;i<(len/2);i++)                    //odd length=>lower bound=>truncate
    {
     if(string[i]!=string[len-i-1])
          palindrome=0;
    }
if(palindrome==0)
               printf("\nThe input string is not palindrome.");

    else
               printf("\nThe input string is palindrome.");
getch();
return 0;
}
```

More String Functions

- *strncpy()* : Copies the left-most *n* characters of the source string to the target string variable. It contains 3 parameters.
- Syntax:

  strncpy(target_string, source_string, n);

- This statement copies the first *n* characters of the source_string into the target_string.
- Note: Since the first *n* characters may not include the terminating null character, we have to place it explicitly in the (n+1)th position as:

  target_string[n]='\0';

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
char target_str[25], source_str[30];
int n;
printf("\nInput string:\t");
gets(source_str);
printf("\nHow many characters to copy:");
scanf("%d", &n);
strncpy(target_str, source_str, n);
target_str[n]='\0';
puts(target_str);
getch();
return 0;
}
```

More String Functions…

- *strncmp()* : Compares the left-most *n* characters of two strings and returns an integer which may be

  (a) 0, if two strings are equal

  (b) negative number, if first string is less than second string (by first unmatched ASCII value from left)

  (c) positive number, otherwise.

- Syntax:

```
strncmp(first_string, second_string, n);
```

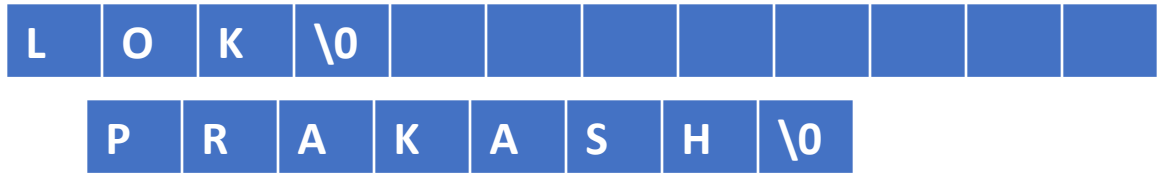
```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
char first_str[25],second_str[30];
int n;
printf("\nInput first string:\t");
gets(first_str);
printf("\nInput second string:\t");
gets(second_str);
n=strncmp(first_str,second_str,5);
printf("%d", n);
getch();
return 0;
}
```

More String Functions…

- *strncat()* : Concatenates the left-most *n* characters of second string to the end of first string.
- Syntax:

  strncat(first_string, second_string, n);

- first_string
- second_string



- strncat(first_string, second_string, 6);

More String Functions…

- *strstr()* : Locates a substring in a string.
- Syntax:

```
strstr(string, substring);
strstr(string, “ABC”);
```

- Here, the function strstr() searches the string “string” to see whether the string “substring” is contained in the “string” or not.
- When no match for the substring is found, it returns NULL.

```
#include<stdio.h>
#include<conio.h>
int main()
{
char string[30],substring[20];
printf("\nInput your string:\t");
gets(string);
printf("\nInput substring you want to search:\t");
gets(substring);
if(strstr(string, substring)==NULL)
        printf("\nSubstring not found.");
else
        printf("\nYour substring %s is contained in %s", substring, string);
getch();
return 0;
}
```

Problems

- Write a program to count the number of words in a sentence.
- Write a program which will read a line and squeeze out all blanks from it and output the line with no blanks.

```
#include <stdio.h>
#include <conio.h>
int main()
{
char sentence[80];
int i, words=0;
printf("Enter text:");
gets(sentence);
for(i=0;sentence[i]!='\0';i++)
 {
   if(sentence[i]==' '||sentence[i]=='\t')
      words++;
 }
words=words+1;          //for counting the word before Enter key
printf("\nThe number of words=%d", words);
getch();
return 0;
}
```

```
#include <stdio.h>
#include <conio.h>
int main()
{
char line[80];
int i, j, k;
printf("\nInput a line of text:\t");
gets(line);
   for(j=0;line[j]!='\0';j++)
    {
    if(line[j]==' ')
     {
            i=j;
            while(line[i]!='\0')
            {
                    line[i]=line[i+1];
                    i++;
                    }
            j--;
            }
   }
puts(line);
getch();
}
```

Arrays of Strings

- String is array of characters.
- Thus an array of string is 2-D array of characters.
- E.g.

```
char names[5][10];
```

- Here, names[5][10] means 5 names having 10 characters each.

```
/*Program to read name of 5 persons using array of strings and display them*/
#include <stdio.h>
#include <conio.h>
int main()
{
char names[5][10];
int i;
printf("\nEnter name of 5 persons:");
for(i=0;i<5;i++)
    scanf("%s", names[i]);

printf("\nThe names are:");
for(i=0;i<5;i++)
   printf("\n%s", names[i]);
getch();
return 0;
}
```

```
/*Program to sort name of 5 persons in ascending order*/
int main()
{
char names[5][10],temp[10];
int i, j;
printf("\nEnter name of 5 persons:");
for(i=0;i<5;i++)
   gets(names[i]);
for(i=0;i<5;i++) {
    for(j=i+1;j<5;j++)   {
    if(strcmp(names[i], names[j])>0)
      {
                strcpy(temp, names[i]);
                strcpy(names[i], names[j]);
                strcpy(names[j], temp);
                }
    }
}
printf("\nNames in ascending order:\n");
for(i=0;i<5;i++)
    puts(names[i]);
getch();
}
```

Thank you !