

Strings and One-Dimensional Character Arrays

CS10003 PROGRAMMING AND DATA STRUCTURES



INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Recap: integer arrays

All the data items constituting the group share the same name

```
int x[10];
```

Individual elements are accessed by specifying the index



x[0] x[1] x[2]

x[9]

X is a 10-element one dimensional array

Can initialize an array during declaration:

```
int x[5] = {76, 99, 0, -3, 8};
```

Character Arrays and Strings

```
char C[8] = { 'a', 'b', 'h', 'i', 'j', 'i', 't', '\0' };
```

C[0] gets the value 'a', C[1] the value 'b', and so on. The last (7th) location receives the null character '\0'

Null-terminated (last character is '\0') character arrays are also called **null-terminated strings** or just **strings**.

Strings can be initialized in an alternative way. The last declaration is equivalent to:

```
char C[8] = "abhijit";
```

C automatically puts the trailing null character at the end if you define a string like this

Note also that for individual characters, C uses single quotes, whereas for strings, it uses double quotes

String literals

String literal values are represented by sequences of characters between double quotes (“

Examples

- “” represents empty string
- “hello”

“a” versus ‘a’

- ‘a’ is a single character value (stored in 1 byte) as the ASCII value for the letter, a
- “a” is an array with two characters, the first is a, the second is the character value \0

Reading strings: %s format

```
int main()
{
    char name[25];
    scanf("%s", name);
    printf("Name = %s \n", name);
    return 0;
}
```

%s reads a string into a character array given the array name or start address.

It ends the string with the special “null” character ‘\0’.

scanf with %s can read a string without any whitespace (blank).

Input assumed to end if whitespace is encountered.

Example: Finding length of a string

```
#define SIZE 25
int main()
{
    int i, length=0;
    char name[SIZE];
    scanf("%s", name);
    printf("Name = %s \n", name);
    for (i=0; name[i]!='\0'; i++)
        length++;
    printf("Length = %d\n", length);
    return 0;
}
```

Note that character strings read in %s format end with '\0'

Output

```
Satyanarayana
Name = Satyanarayana
Length = 13
```

Example: Counting the number of a's

```
#define SIZE 25
int main()
{
    int i, count=0;
    char name[SIZE];
    scanf("%s", name);
    printf("Name = %s \n", name);
    for (i=0; name[i]!='\0'; i++)
        if (name[i] == 'a') count++;
    printf("Total a's = %d\n", count);
    return 0;
}
```

Output

Satyanarayana

Name = Satyanarayana

Total a's = 6

**Note that character strings read
in %s format end with '\0'**

Example: Palindrome Checking

```
int main()
{
    int i, flag, count=0;
    char name[25];
    scanf("%s", name);           /* Read Name */
    for (i=0; name[i]!='\0'; i++); /* Find Length of String */
    count=i; flag = 0;
    /* Loop below checks for palindrome by comparison */
    for(i=0; i<count; i++)
        if (name[i]!=name[count-i-1])
            flag = 1;
    if (flag ==0) printf ("%s is a Palindrome\n", name);
    else printf ("%s is NOT a Palindrome\n", name);
    return 0;
}
```

Is there scope for making it more efficient?

Counting Vowels of a String

```
#include<stdio.h>
main()
{
    char A[40], B[5]={'a', 'e', 'i', 'o', 'u'};
    int i, j, len, C[5]= {0,0,0,0,0};
    scanf("%s", A);
    printf("A = %s \n", A);
    for (i=0; A[i]!='\0'; i++);
    len = i;
    printf("Length = %d\n", len);
    for(i=0; i<len; i++){
        for(j=0; j<5;j++)
            if(A[i]== B[j]) C[j]++;
    }
    for (j=0;j<5;j++)
        printf("Number of %c = %d \n", B[j], C[j]);
}
```

```
thequickbrownfoxjumpsoverthelazydog
A = thequickbrownfoxjumpsoverthelazydog
Length = 35
Number of a = 1
Number of e = 3
Number of i = 1
Number of o = 4
Number of u = 2
```

Reading Strings with Blanks

In many applications, we need to read in an entire line of text (including blank spaces).

But scanf with %s cannot input a string having whitespace within it

One way to enter a string having whitespace within it:

```
char    line[81], ch;
int     c=0;
:
:
do
{
    ch = getchar();
    line[c] = ch;
    c++;
}
while  (ch != '\n');

c = c - 1;
line[c] = '\0';
```

Read characters
until CR ('\n') is
encountered

Make it a valid string

Reading Strings with Blanks: Alternate approach

In many applications, we need to read in an entire line of text (including blank spaces).

But scanf with %s cannot input a string having whitespace within it

Another way to enter a string having whitespace within it:

```
char line[81];  
:  
:  
scanf ("%[^\\n]", line);
```

Reads a string containing any characters until a newline character is received

Pattern Matching

```
#include<stdio.h>
main()
{
char S[20], P[20];
int i,j, k, flag;
printf("Enter String and Pattern:\n");
scanf("%s%s", S, P);
printf("S = %s,P = %s \n", S, P);
k =0;
for (i=0; S[i] != '\0'; i++){
    flag = 1;
    for(j=0;P[j]!='\0'; j++)
        if (S[i+j]!= P[j]) {flag = 0; break;}
    if (flag == 1) k++;
}
printf("Number of Matches = %d \n", k);
}
```

Enter String and Pattern:
abababababab
aba
S = abababababab,P = aba
Number of Matches = 5

Enter String and Pattern:
abababababab
ababab
S = abababababab,P = ababab
Number of Matches = 4

String library functions

Library Functions

- Set of functions already written for you, and bundled in a “library”
 - Example: `printf`, `scanf`, `getchar`,
- C library provides a large number of functions for many things
- Already seen math library functions earlier
- Will look at string library functions

String Library Functions

String library functions

- perform common operations on null terminated strings
- Must include a special header file
`#include <string.h>`

Note:

We have discussed -- when an integer / float array is sent to a function as an argument, it is required to send an indicator of the size or number of valid elements in the array as another argument

When a string is sent to a function as an argument, no need to send any separate indicator of the size, since the function can utilize the '\0' to identify the end of the string.

Common string library functions

strlen – returns the length of a string

strcmp – compares two strings (lexicographic)

strcat – concatenates two strings

strcpy – copy one string to another

Many others, but these are the ones you will know in this course

strcpy()

Works very much like a string assignment operator.

```
char* strcpy (char *str1, char *str2);
```

- Assigns the contents of **str2** to **str1**.
- Returns address of the destination string.

Examples:

```
strcpy(city, "Calcutta");  
strcpy(city, mycity);
```

Warning:

- Assignment operator do not work for strings.

```
city = "Calcutta"; ➔ INVALID
```

strlen()

Counts and returns the number of characters in a string.

```
int strlen (char *str);
```

Example:

```
len = strlen(str) ;
```

```
/* str should be null-terminated. Returns an integer */
```

- The null character ('\0') at the end is not counted.
- Counting ends at the first null character.

strcmp()

Compares two character strings (lexicographic comparison).

```
int strcmp(char *str1, char *str2);
```

- Returns 0 if the two strings are equal, < 0 if first string is lexicographically less than the second string, > 0 if the first string is lexicographically greater than the second string

Examples:

```
if (strcmp(city, "Delhi") == 0)
{ ..... }
```

```
if (strcmp(city1, city2) != 0)
{ ..... }
```

Example

```
int main()
{
    char A[20], B[20];
    int n, m, val;
    scanf("%s%s", A, B);
    n = strlen(A);
    m = strlen(B);
    printf("The lengths of the strings are %d and %d\n", n, m);
    val = strcmp(A, B);
    if (val == 0)
        printf("The strings are the same\n");
    else if (val < 0)
        printf("%s is smaller than %s\n", A, B);
    else
        printf("%s is smaller than %s\n", A, B);
}
```

strcat()

Joins or concatenates two strings together

```
char* strcat (char *str1, char *str2);
```

- **str2** is appended to the end of **str1**.
- The null character at the end of **str1** is removed, and **str2** is joined at that point.
- **str1** should have enough space -- Programmer's responsibility

Example:

```
strcpy(name1, "Amit ");
```

```
strcpy(name2, "Roy");
```

```
strcat(name1, name2);
```

A	m	i	t		'\0'			
---	---	---	---	--	------	--	--	--

R	o	y	'\0'
---	---	---	------

A	m	i	t		R	o	y	'\0'
---	---	---	---	--	---	---	---	------