Computer Programming & Problem Solving

CS100

Mrs Sanga G. Chaki
Department of Computer Science and Engineering
National Institute of Technology, Goa
May, 2023

Strings

1. Character arrays used to store a collection of characters
2. Used to manipulate text such as words and sentences.
3. A string always terminates with a NULL character –
    a) Only way to know where the string ends
    b) ASCII value = 0
    c) Represented as = ‘\0’
4. Collection of characters vs Strings

Strings - Initialization

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

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

1. C[0] stores the value 'a', C[1] the value 'b', and so on.
2. The trailing null character is missing in the second method.
    a) C automatically puts it at the end if you define it like this
3. Note
    a) for individual characters, C uses single quotes,
    b) for strings, it uses double quotes

Access string elements – Method 1

```
main( )
{
      char  name[ ] = "Klinsman" ;
      int  i = 0 ;

      while ( i <= 7 )
      {
            printf ( "%c", name[i] ) ;
            i++ ;
       }

   }
```

Access string elements – Method 2

```
main( )
{
      char  name[ ] = "Klinsman" ;
      int  i = 0 ;

      while ( name[i] != `\0' )
      {
            printf ( "%c", name[i] ) ;
            i++ ;
      }
}
```

1. Works even if we do not know the length of the string.

Access string elements – Method 3

```
main( )
{
      char  name[ ] = "Klinsman" ;
      char  *ptr ;

      ptr = name ;  /* store base address of string */

      while ( *ptr != `\0' )
      {
            printf ( "%c", *ptr ) ;
            ptr++ ;
      }
}
```

1. Using Pointers

Access string elements – Method 4

```
main( )
{
        char  name[ ] = "Klinsman" ;
        printf ( "%s", name ) ;
}
```

1. **Using printf()**
2. **%s is the format specifier for the string**

Receive a string from user

```
main( )
{
      char  name[25] ;

      printf ( "Enter your name " ) ;
      scanf ( "%s", name ) ;
      printf ( "Hello %s!", name ) ;
}
```

1. **Using scanf()**
2. **%s is the format specifier for the string**
3. **Note: No & in scanf(). Why?**
   a) **we are passing the base address of the array to the scanf( )**
   b) **scanf( ) fills in the characters from keyboard until the enter key is hit.**

Receive a string from user - Example

```c
#include <stdio.h>
int main() {
    char a[25];
    int c =0; //counter for number of letters in string
    printf("Enter string ");
    scanf("%s",a);
    printf("\n%s",a);
    for(int i=0; a[i]!='\0'; i++){
        printf("\n%c",a[i]);
        c++;
    }
    printf("\nNumber of letters in this string = %d",c);
    return 0;
}
```

Receive a string from user - Example

Output

```
/tmp/vW4n3oWg1A.o
Enter string NITGoa
NITGoa
N
I
T
G
o
a
Number of letters in this string = 6
```

Some Limitations

1. The length of the string should not exceed the dimension of the character array. – Why?
   a) Something might be overwritten
   b) C compiler doesn't perform bounds checking
2. scanf( ) is not capable of receiving multi-word strings. So we use the following library functions from stdio.h
   a) gets()
   b) puts()

gets() and puts()

```
main( )
{
    char  name[25] ;

    printf ( "Enter your full name " ) ;
    gets ( name ) ;
    puts ( "Hello!" ) ;
    puts ( name ) ;
}
```

1. **Why two puts()? – It can display only one string at a time.**
2. **gets() can receive multi-word strings, though it can receive only 1 string at a time.**

Some Differences

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

Standard String Library Functions

1. There exists a set of C library functions for character string manipulation.
    a) strcpy :: string copy
    b) strlen :: string length
    c) strcmp :: string comparison
    d) strcat :: string concatenation
2. It is required to add the line
    a) #include <string.h>

strlen( )

```
main( )
{
        char  arr[ ] = "Bamboozled" ;
        int  len1, len2 ;

        len1 = strlen ( arr ) ;
        len2 = strlen ( "Humpty Dumpty" ) ;

        printf ( "\nstring = %s length = %d", arr, len1 ) ;
        printf ( "\nstring = %s length = %d", "Humpty Dumpty", len2 ) ;
}
```

1. **Counts the number of characters present in a string**

1. **What are the outputs?**
2. **Note – It doesn't count the Null character.**

strcpy( )

```
main( )
{
      char  source[ ] = "Sayonara" ;

      char  target[20] ;

      strcpy ( target, source ) ;
      printf ( "\nsource string = %s", source ) ;
      printf ( "\ntarget string = %s", target ) ;
}
```

1. **Copies the contents of one string into another**

1. **What are the outputs?**

strcat( )

```
main( )
{
        char  source[ ] = "Folks!" ;
        char  target[30] = "Hello" ;

        strcat ( target, source ) ;
        printf ( "\nsource string = %s", source ) ;
        printf ( "\ntarget string = %s", target ) ;
}
```

1. **What are the outputs?**

1. **Concatenates the source string at the end of the target string**

Example

```c
// Online C compiler to run C program on
#include <stdio.h>
#include<string.h>
int main() {
    char str1[7]="NIT";
    char str2[7]="GOA";
    char c;
    printf("%s\n",str1);
    printf("%s\n",str2);
    printf("%d \n",strlen(str1));
    printf("%d \n",strlen(str2));
    printf("%s \n",strcat(str1,str2));
    printf("%s \n",strcpy(str2,str1));
    int i;
    for(i=0;i<7;i++){
        printf("%c ",str1[i]);
        printf("%d ",str1[i]);
    }

    return 0;
}
```

Output

```
/tmp/Ub7JCbNHP6.o
NIT
GOA
3
3
NITGOA
NITGOA
N 78 I 73 T 84 G 71 O 79 A 65 · 0
```

strcmp( )

1. Compares two strings to find out whether they are same or different.
2. Strings are compared character by character until there is a mismatch or end of one of the strings.
3. If the two strings are identical, strcmp( ) returns a value zero.
4. If they’re not, it returns the numeric difference between the ASCII values of the first non-matching pairs of characters.

strcmp( )

strcmp examples:

```
strcmp("hello","hello")              -- returns 0
strcmp("yello","hello")              -- returns value > 0
strcmp("Hello","hello")              -- returns value < 0
strcmp("hello","hello there")        -- returns value < 0
strcmp("some diff","some dift")      -- returns value < 0
```

strcmp( )

```c
#include <stdio.h>
#include<string.h>
int main() {
    char a[] = "NITGoa";
    char b[] = "NITGoa";
    char c[] = "NITgoa";
    char d[] = "nITGoa";
    char e[] = "N";
    printf("%d\n", strcmp(a,b));
    printf("%d\n", strcmp(a,c));
    printf("%d\n", strcmp(a,d));
    printf("%d\n", strcmp(d,a));
    printf("%d\n", strcmp(a,e));
    return 0;
}
```

Output

```
/tmp/RtxcfHRDqo.o
0
-32
-32
32
73
```