STRINGS

The way a group of integers can be stored in a integer array, similarly a group of characters can be stored in a character arrays are many a time also called **Strings**.

Character arrays or strings are used by programming languages to manipulate text, such as words and sentences.

A string constant is a one dimensional array of characters terminated by a null ('\0').

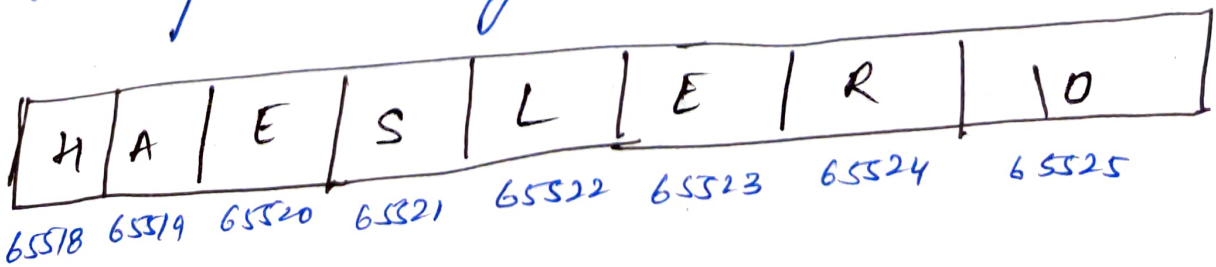
```
char name[] = {'H', 'A', 'E', 'S', 'L', 'E', 'R', '\0'};
```

Each character in the array occupied 1 byte of memory and the last character is always '\0'.

Note: '\0' and '0' are not same. ASCII value of '\0' is 0 whereas ASCII value of '0' is 48.

The terminating null ('\0') is important because it is the only way the functions that work with the string can know where the string ends.

If a string is not terminated by '\0' is not really a string but merely a collection of characters.



```
char name[] = "HMESLER";
```

/* Program to demonstrate printing of a string */

```
#include<stdio.h>
int main()
{
    char name[] = "klinsman";
    int i=0;
    while(i<=7)
    {
        printf("%c", name[i]);
        i++;
    }
    printf("\n");
    return 0;
}
```

O/P

```
klinsman
```

/* program in C to print character array using while loop without ~~using~~ using final value.

```
#include<stdio.h>
int main()
{
    char name[] = "klinsman";
    int i=0;
    while(name[i] != '\0')
    {
```

```
        printf("%c", name[i]);
        i++;
    }
    printf("\n");
    return 0;
}
```

o/p

```
klinsman
```

/* Write a program printing a string that uses a pointer to access the array elements. */

```
#include<stdio.h>
int main()
{
    char name[] = "klinsman";
    char *ptr;
    ptr = name;        /* Store base address of a string */
    while (*ptr != '\0')
    {
        printf("%c", *ptr); // *ptr would
                            // yield the value at
                            // this address.
        ptr++;
    }
    printf("\n");
    return 0;
}
```

Note:

```
char *ptr;
char name[] = "Prasanna";
ptr = name;
```

The base address of the zeroth element of the array is stored in the variable ptr.

```
ptr++;
```

when ptr is incremented to point to the next character in the string. This derives from two facts:
array elements are stored in contiguous memory locations and on incrementing a pointer, it points immediately next location of its type. This process is carried out until ptr points to the last character in the string i.e '\0'.

In fact, the character array elements are accessed exactly in the same way as the elements of an integer array.

```
name[i]
*(name+i)
*(i+name)
i[name]
```

* Use format specification for printing out a (3)
string.

```
#include<stdio.h>
int main()
{
    char name[] = "klinsman";
    printf("%s", name);
}
```

* %s used in printf() is a format specification for printing out a string.

/* program to receive a string from the keyboard */

```
#include<stdio.h>
int main()
{
    char name[25];
    printf("Enter your name ");
    scanf("%s", &name);
    printf("Hello %s!\n", name);
    return 0;
}
```

O/P

```
Enter your name Debashish
Hello Debashish!
```

Important points

1) The length of the string should not exceed the dimension of the character array.

2) Scanf() is not capable of receiving multi-word strings. Therefore, names such as 'Debashish Roy' would be unacceptable.
This limitation is removed by using gets() and puts() functions.

```
#include<stdio.h>
int main()
{
    char name[25];
    printf("Enter your full name:");
    gets(name);
    puts("Hello!");
    puts(name);
    return 0;
}
```

O/P

```
Enter your full name: Debashish Roy
Hello!
Debashish Roy
```

Standard Library String functions

With every C compiler, a large set of useful string handling library functions are provided.

	function	Use
1)	strlen	finds length of a string
2)	strlwr	Converts a string to lowercase.
3)	strupr	Converts a string to uppercase
4)	strcat	Appends one string at the end of another.
5)	strcpy	copies a string into another.
6)	strcmp	Compares two strings
7)	strdup	Duplicates a string
8)	strrev	Reverse a string

strlen

This function counts the number of characters present in a string.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char a[] = "Bamboozled";
    int len1, len2;
    len1 = strlen(a);
    len2 = strlen("Humpty Dumpty");
    printf("string=%s length=%d\n", a, len1);
    printf("string=%s length=%d\n", "Humpty Dumpty",
                                    len2);
```

```
    return 0;
}
```

o/p

```
String = Bamboozled Length = 10
String = Humpty Dumpty Length = 13.
```

2) Strcpy

This function copies the contents of one string into another. The base address of the source and target strings should be supplied to the function.

```
#include<stdio.h>
#include<string.h>
int main()
{
char source[] = "Sayonara";
  char target[20];
  strcpy(target, source);
  printf("source string = %s\n", source);
  printf("target string = %s\n", target);
  return 0;
}
```

o/p

```
source string = sayonara
target string = sayonara
```

3) strcat

This function concatenates the source string at the end of the target string.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char source[] = "folks!";
    char target[30] = "Hello";
    strcat(target, source);
    printf("source string=%s\n", source);
    printf("target string=%s\n", target);
    return 0;
}
```

O/P

```
source string = folks!
target string = Hellofolks!
```

(4)

Strcmp is a function which compares two strings to find out whether they are same or different. The two strings are compared character by character until there is a mismatch or end of one of the strings is reached, whichever occurs first.

If the two strings are identical, strcmp() returns a value 0. If they are not, it returns the numeric difference between the ASCII value of the first non-matching pair of characters.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char string1[] = "Jerry";
    char string2[] = "Ferry";
    int i, j, k;
    i = strcmp(string2, "Jerry");
    j = strcmp(string2, string2);
    k = strcmp(string1, "Jerry boy");
    printf("%d %d %d\n", i, j, k);
    return 0;
}
```

O/P

0	4	-32

(1) In the first call to strcmp(), the two strings are identical the value returned by strcmp() is zero.

(2) In the second call, the first character of "Jerry" doesn't match with the first character of "Ferry" and the result is 4, which is the numeric difference between ASCII value of 'J' and ASCII value of 'F'.

(3) In the third call to strcmp(), "Jerry" does not match with blank in "Jerry boy". The value returned is -32, which is the value of null character minus the ASCII value of space ie '\0' minus ' ' which is equal to -32.