

Strings

- A string of characters is a 1D array of characters
- ASCII code (1 byte) of each character element is stored in consecutive memory locations
- String is terminated by the null character '\0' (ASCII value 0).
- The null string (length zero) is the null character only

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

\0 is used to indicate termination of a string



Length is 7, but the array is [0 ... 7]

Reading and printing strings

- A string can be read using `scanf`
 - Format conversion specifier for a sequence of non-white-space characters is `%s`.

E.g. `scanf("%s", a)` will store only “Comp” for user input “Comp Sc”

➤ And `%[^\n]` for strings with white-space included

E.g. `scanf("%[^\n]", a)` will store only “Comp Sc” for user input “Comp Sc”

- A string can be printed using `printf()` with `%s`

Library string.h

- Library functions are available to manipulate strings.
- A list of commonly used string manipulation functions:
 - `strcpy()` - copy a string
 - `strcat()` - concatenate two strings
 - `strlen()` - get string length
 - `strcmp()` - compare two strings

strcpy - copy a string

```
#include <stdio.h>
#include <string.h>
int main () {
    char src[40];
    char dest[100];

    printf("Enter source string:");
    scanf("%[^\n]", src);
    strcpy(dest, src);

    printf("Destination string : %s\n", dest);

    return(0);
}
```

```
Enter source string:How are you?
Destination string : How are you?
```

strcat - concatenate two strings

- `strcat` joins two strings together.

`strcat(string1, string2)`

- `string2` is appended to `string1`
- `string2` remains unchanged

```
#include <stdio.h>
#include <string.h>
int main () {
    char str[100] = "Hello ";
    strcat(str, "World!");

    printf("New string : %s\n", str);

    return(0);
}
```

```
New string : Hello World!
```

strlen – length of a string

- Returns the number of characters in the input string
`n = strlen(string)`

```
#include <stdio.h>
#include <string.h>
int main () {
    char str[100];
    int n;

    printf("Enter source string:");
    scanf("%[^\n]", str);
    n = strlen(str);
    printf("Length is: %d\n", n);

    return(0);
}
```

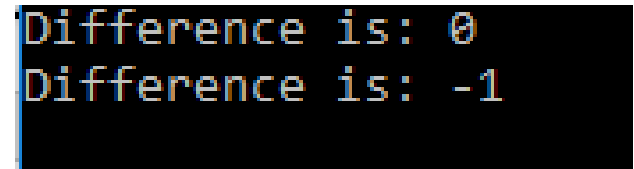
```
Enter source string:Hello World
Length is: 11
```

strcmp – compares two strings

`n = strcmp(string1, string2)`

- If the strings are equal then returns 0
- Otherwise, returns the numeric difference between the first non matching characters

```
int main () {  
    char str1[100] = "Hello World!";  
    char str2[100] = "Hello World!";  
    char str3[100] = "Hello Wosld!";  
  
    int n;  
  
    n = strcmp(str1, str2);  
    printf("Difference is: %d\n", n);  
  
    n = strcmp(str1, str3);  
    printf("Difference is: %d\n", n);  
  
    return(0);  
}
```



```
Difference is: 0  
Difference is: -1
```

Space issues

- Need to ensure that sufficient memory is allocated at run-time for storing the string
 - Not automatically done by the compiler
 - No checks are done at compile-time or run-time
 - Common source of errors like accidental overwriting of other information
 - Has been exploited to achieve execution of code supplied by attackers
 - Problem is called Buffer Overflow
-
- Very different in Java: Memory allocated by compiler, and length of strings checked at run time => Buffer overflow impossible.
 - Will discuss prevention mechanisms next week.