Strings

↓

A character array terminated by a '\0' (null character)

null character denotes string termination

EXAMPLE

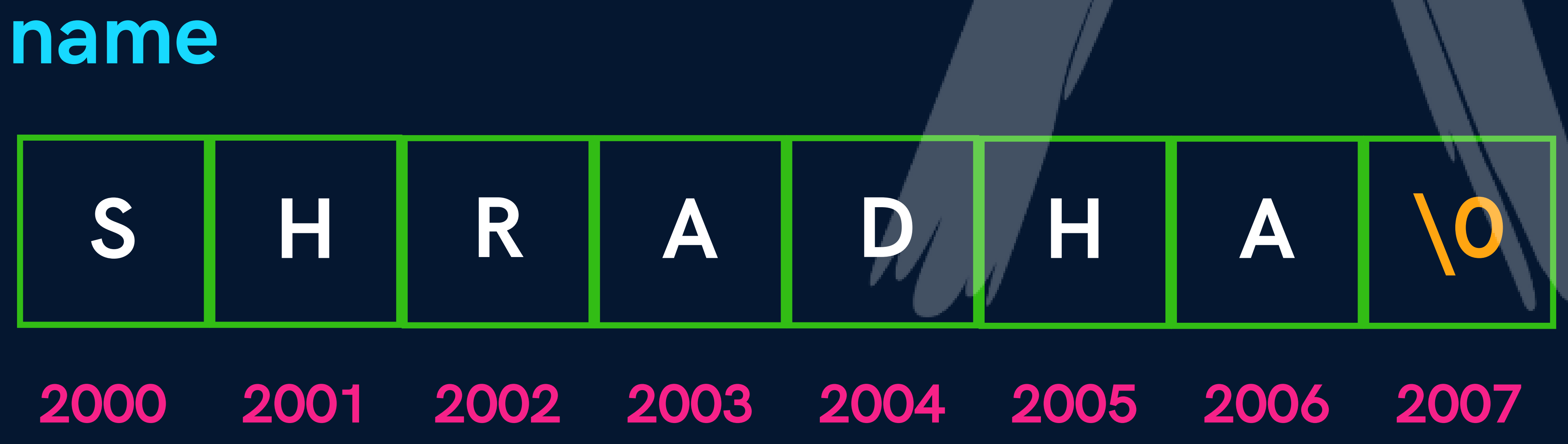
```
char name[ ] = {'S', 'H', 'R', 'A', 'D', 'H', 'A', '\0'};

char class[ ] = {'A', 'P', 'N', 'A', ' ', 'C', 'O', 'L', 'L', 'E', 'G', 'E', '\0'};
```

Initialising Strings

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

char name[ ] = "SHRADHA";



char class[ ] = {'A', 'P', 'N', 'A', ' ', 'C', 'O', 'L', 'L', 'E', 'G', 'E', '\0'};

char class[ ] = "APNA COLLEGE";
```

What Happens in Memory?

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

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



String Format Specifier

↓

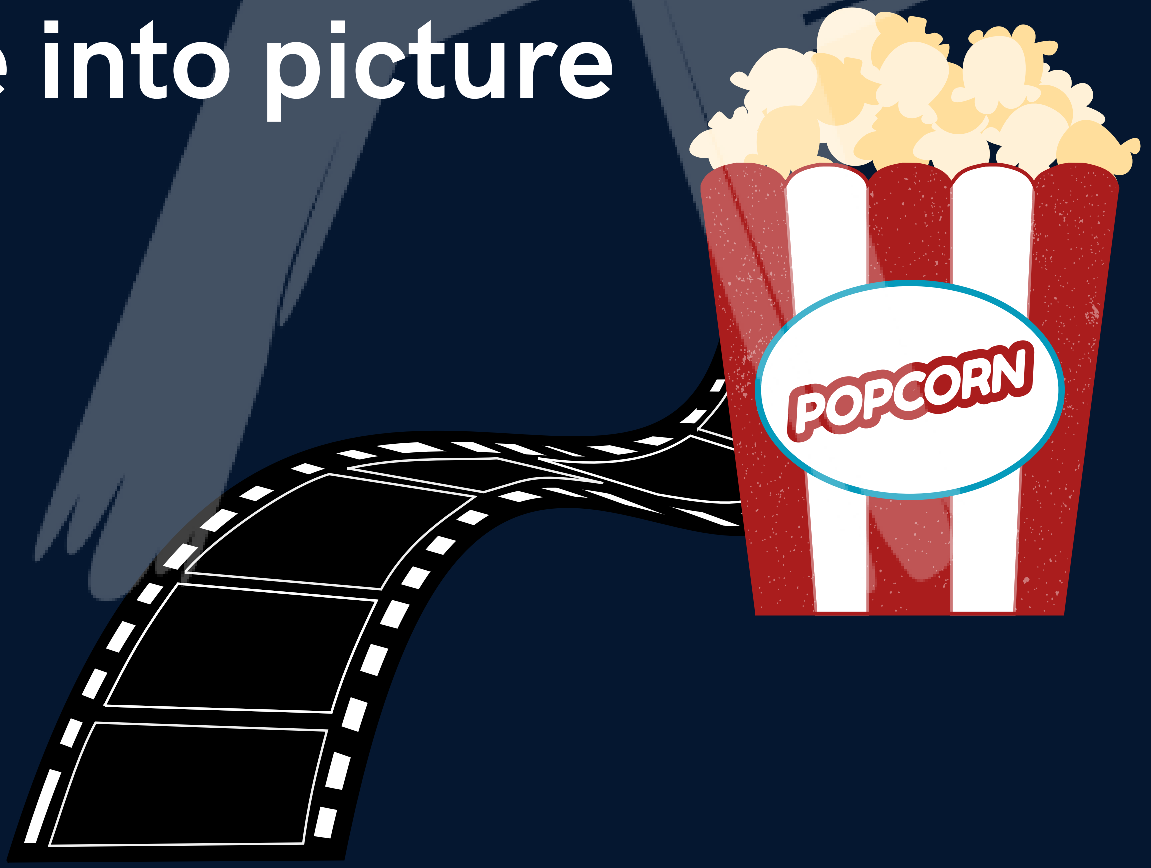
"%s"

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

printf("%s", name);
```

IMPORTANT

scanf( ) cannot input multi-word strings with spaces

Here,
gets( ) & puts( ) come into picture



String Functions

gets(str) $\longrightarrow$ Dangerous & Outdated

input a string (even multiword)

puts(str)

output a string

fgets(str, n, file)

stops when n-1 chars input or new line is entered

String using Pointers

```
char *str = "Hello World";
```

Store string in memory & the assigned address is stored in the char pointer 'str'

```
char *str = "Hello World";    //can be reinitialized
```

```
char str[ ] = "Hello World";
//cannot be reinitialized
```

Standard Library Functions

↓

<string.h>

1 strlen(str)

count number of characters excluding '\0'

Standard Library Functions

$\downarrow$

<string.h>

2 strcpy(newStr, oldStr)

copies value of old string to new string

Standard Library Functions

$\downarrow$

<string.h>

3 strcat(firstStr, secStr)

concatenates first string with second string

firstStr should be large enough

Standard Library Functions

$\downarrow$

<string.h>

4 strcpm(firstStr, secStr)

Compares 2 strings & returns a value

0 -> string equal

positive -> first > second (ASCII)

negative -> first < second (ASCII)