Syrian Private University

Introduction to Algorithms and Programming

Instructor: Dr. Mouhib Alnoukari

7 Strings

Outline

- Declare a String
- Input a String
- Assign a String
- String functions
- Character type operations

Print a character Array

Program:

```
// character array
char c[10] = {'E', 'E', '/','C', 'p', 'r','E'};
for(i=0; i< 7; i++)
{
  printf("%c", c[i]);
}
```

Output: EE/CprE

Print another way

Program:

```
// character array
char c[10] = {'E', 'E', '/','C', 'p', 'r','E'};

printf("%s", c); //%s to print a string
```

Output: EE/CprE

Declare another way

```
Program:
// define character array with a
  string
char c[10] = “EE/CprE”;
printf(“%s”, c); //%s to print a
  string
```

_____

Output: EE/CprE

Character vs. String

- Do NOT confuse strings with individual characters

```
'E' is a character

"E" is a string
```

- initialize a character array with
  1) Array of characters
     `{'E','E','/','C','p','r','E'}`
  2) A String
     `"EE/CprE"//easier to type`

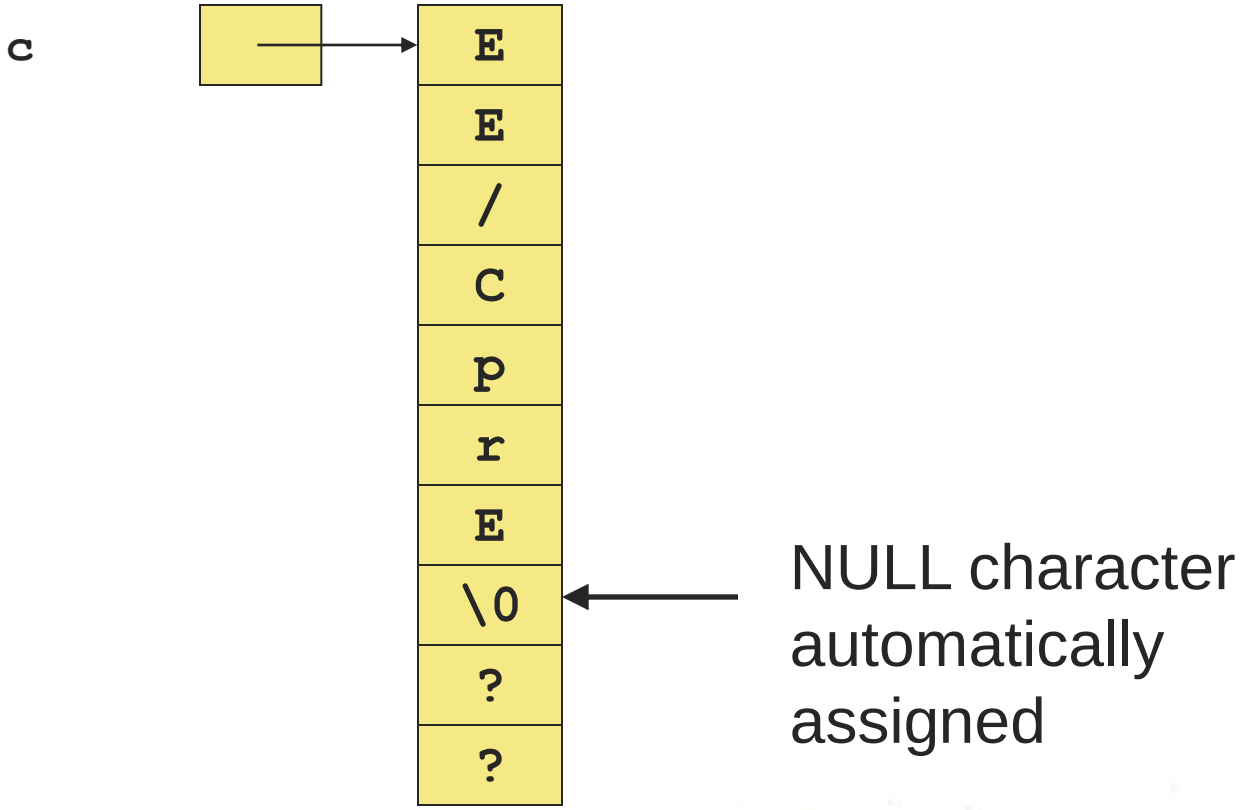
Declaring strings

```
char c[10] = "EE/CprE"; //only 7
  chars
```

- What about the uninitialized characters in the string?

Array contents

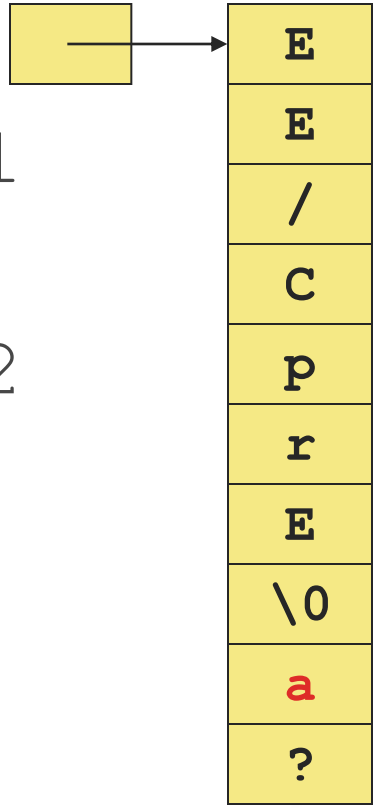
- Contents of the *c* array



Array Contents

Program:

```
// character array
char c[10] = "EE/CprE";c
printf("%s", c); //output 1
c[8] = 'a';
printf("%s", c); //output 2
```



Output 1: EE/CprE

Output 2: EE/CprE

Strings, what's happening

- Prints until NULL character is reached
  - leave room for it!

```
Program:
// array size should be >= 8
char c[7] = "EE/CprE";
printf("%s", c); //%s to print a string
```

```
Output:
```

Strings

- *Length* is determined by first NULL in the string
- Most string functions in C add NULL automatically
- Array of strings is a double array of characters
  - From the book:

```
char month[12][10] = {"January", "February", … ,
   "December"};
```

Input a String

Program:

```
char c[N];
scanf("%s", c);  //no & symbol is required
printf("%s", c);
```

Input: "EE CprE"

Output: EE //input separated by white space

Input a String

- *gets*
  - Get a string from user input
  - reads until enter is pressed

```
char c[N];
gets(c);
printf("%s\n", c);
```

```
Input: "EE CprE"
Output: EE CprE
```

Assign value to a String

- Cannot use = operator in C to assign a String

```
Program:
// character array
char c[N];
c = “Monday”; //will NOT work
```

Assign value to a String

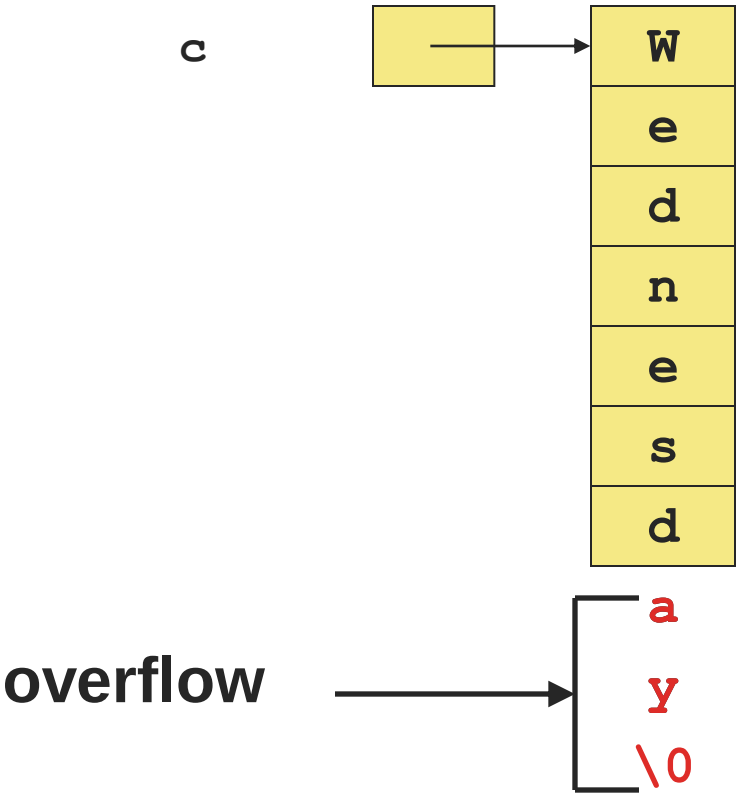
- Use the String function *strcpy* in string.h
  - Copies a string into a destination string

```
#include <string.h>
…
char c[N];
char tomorrow[] = "Tuesday";
…
strcpy(c, "Monday"); //c is the destination
…
strcpy(c, tomorrow); //another assignment
```

Assign value to a String

- Watch out for overflow (bad)

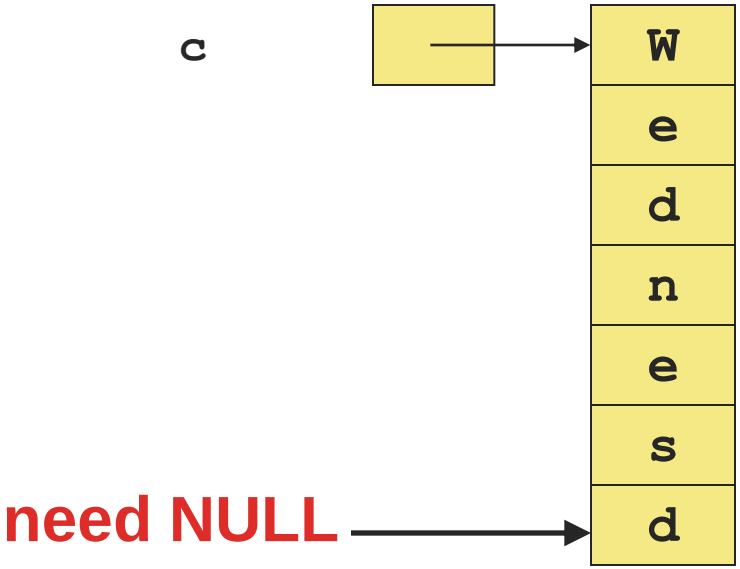
```
#include <string.h>
…
char c[7];
strcpy(c, "Wednesday");
```



Assign value to a String

- Better to use *strncpy*
  - copies up to *n* characters from the source

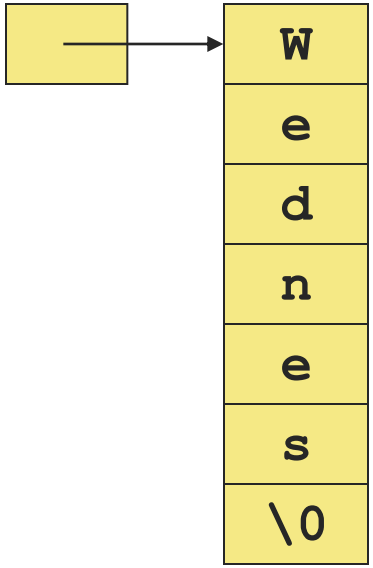
```
#include <string.h>
…
char c[7];
strncpy(c, "Wednesday", 7);
```



Assign value to a String

- Better to use *strncpy*
  - assign NULL to the end afterword

```
#include <string.h>
…
char c[7];
strncpy(c, "Wednesday", 6);
c[6] = '\0';
//OR
c[6] = NULL; //NULL and '\0' are the same
```



String Functions <string.h>

- *strcmp* or *strncmp*
  - Compares two strings (good for sorting)

```
strcmp("Saturday", "Sunday");
   //answer is -1

strncmp("way", "wash", 2);
   //answer is 0
```

- *strlen*
  - Returns the number of characters in *"Saturday"*

```
strlen("Saturday")
   //answer is 8
```

String Functions <string.h>

- *strcat*
  - Concatenate two strings (good for sorting)

```
char a[N] = "Hello ";
char b[N] = "World!\n";
strcat(a, b);

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

```
Output: Hello World!
```

Character Operations <ctype.h>

- `isalpha`
  - is the character a letter of the alphabet?
- `isdigit`
  - Is the character a number?
- `islower, isupper`
  - Checks the case of the letter
- `ispunct`
- `isspace`