

#### Lecture #5b

# Arrays, Strings and Structures





#### Questions?

# IMPORTANT: DO NOT SCAN THE QR CODE IN THE VIDEO RECORDINGS. THEY NO LONGER WORK

Ask at

https://sets.netlify.app/module/676ca3a07d7f5ffc1741dc65

#### OR

Scan and ask your questions here! (May be obscured in some slides)



#### 3. Strings (1/2)

```
#include <stdio.h>

void modifyArray(char [], int);
void printArray(char [], int);

int main(void) {
   char chars[4] = {'C', 'h', 'a', 'r'};
   modifyArray(chars, 4);
   printArray(chars, 4);
   return 0;
}
```

#### Array of characters

The following code is very similar to ArrayModify.c. What does it do? What is the output?

Dibs

```
void modifyArray(char arr[], int size) {
  int i;
  for (i=0; i<size; i++) {
    arr[i]++;
  }
  for (i=0;
  printf</pre>
```

```
void printArray(char arr[], int size) {
  int i;

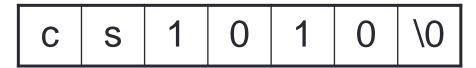
for (i=0; i<size; i++) {
    printf("%c", arr[i]);
  }
  printf("\n");
}</pre>
```



### 3. Strings (2/2)

- We can turn an array of characters into a string by adding a null character '\0' at the end of the array
- A string is an array of characters, terminated by a null character '\0' (which has an ASCII value of zero)
- We can use string functions (include <string.h>) to manipulate strings.

Example:





#### 3.1 Strings: Basic

Declaration of an array of characters

```
char str[6];
```

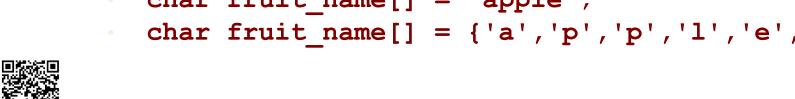
Assigning character to an element of an array of

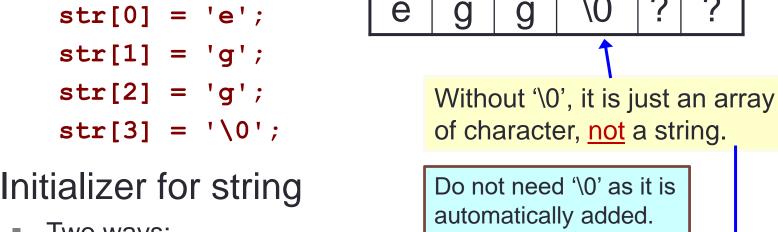
characters

```
str[0] = 'e';
str[1] = 'g';
str[2] = 'g';
```

- Initializer for string
  - Two ways:

```
char fruit name[] = "apple";
char fruit_name[] = {'a','p','p','l','e','\0'};
```





### 3.2 Strings: I/O (1/3)

Read string from stdin (keyboard)

Print string to stdout (monitor)

```
puts(str); // terminates with newline
printf("%s\n", str);
```

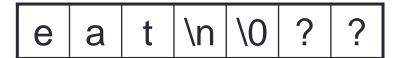
Note: There is another function gets(str) to read a string interactively. However, due to security reason, we avoid it and use fgets() function instead.



# 3.2 Strings: I/O (2/3)

- fgets()
  - On interactive input, fgets() also reads in the newline character

User input: eat



 Hence, we may need to replace it with '\0' if necessary

```
fgets(str, size, stdin);
len = strlen(str);
if (str[len - 1] == '\n')
    str[len - 1] = '\0';
```



# 3.2 Strings: I/O (3/3)

```
StringIO1.c
```

```
#include <stdio.h>
#define LENGTH 10
Int main(void) {
   char str[LENGTH];

   printf("Enter string (at most %d characters): ", LENGTH-1);
   scanf("%s", str);
   printf("str = %s\n", str);
   printf("str = %s\n", str);
   return 0;
}
```

```
#include <stdio.h>
#define LENGTH 10

Output:
str = My book

int main(void) {
   char str[LENGTH];

   printf("Enter string (at most %d characters): ", LENGTH-1);
   fgets(str, LENGTH, stdin);
   printf("str = ");
   puts(str);
   return 0;
Note that puts(str) adds
a newline automatically.
```

#### 3.3 Example: Remove Vowels (1/2)

- Write a program RemoveVowels.c to remove all vowels in a given input string.
- Assume the input string has at most 100 characters.
- Sample run:

Enter a string: How HAVE you been, James?

Changed string: Hw HV y bn, Jms?



#### 3.3 Example: Remove Vowels (2/2)

```
RemoveVowels.c
#include <stdio.h>
                               Need to include <string.h>
#include <string.h>
                               to use string functions such
#include <ctype.h>
                               as strlen().
                                            Need to include <ctype.h> to
int main(void) {
                                             use character functions such
   int i, len, count = 0;
   char str[101], newstr[101];
                                             as toupper().
   printf("Enter a string (at most 100 characters): ");
   fgets(str, 101, stdin); //what happens if you use scanf() here?
   len = strlen(str); // strlen() returns number of char in string
   if (str[len - 1] == '\n')
      str[len - 1] = ' \ 0';
   len = strlen(str); // check length again
   for (i=0; i<len; i++) {
      switch (toupper(str[i])) {
         case 'A': case 'E':
         case 'I': case 'O': case 'U': break;
         default: newstr[count++] = str[i];
   newstr[count] = ' \setminus 0';
   printf("New string: %s\n", newstr);
   return 0;
```

### 3.4 String Functions (1/2)

- C provides a library of string functions
  - Must include <string.h>
  - Here are a few commonly used string functions
- strlen(s)
  - Return the number of characters in s
- strcmp(s1, s2)
  - Compare the ASCII values of the corresponding characters in strings s1 and s2.
  - Return
    - a negative integer if s1 is lexicographically less than s2, or
    - a positive integer if s1 is lexicographically greater than s2, or
    - 0 if s1 and s2 are equal.
- strncmp(s1, s2, n)
  - Compare first n characters of s1 and s2.



### 3.4 String Functions (2/2)

- strcpy(s1, s2)
  - Copy the string pointed to by s2 into array pointed to by s1.
  - Function returns s1.
  - Example:

```
char name[10];

M a t t h e w 0 ? ?

strcpy(name, "Matthew");
```

The following assignment statement <u>does not work</u>:

```
name = "Matthew";
```

What happens when string to be copied is too long?

```
strcpy(name, "A very long name");
```



- strncpy(s1, s2, n)
  - Copy first n characters of string pointed to by s2 to s1.



# 3.5 Importance of '\0' in a String (1/2)

- To be treated as a string, the array of characters must be terminated with the null character '\0'.
- Otherwise, string functions will not work properly on it.
- For instance, the printf("%s", str) statement will print until it encounters a null character in str.
- Likewise, strlen(str) will count the number of characters up to (but not including) the null character.
- In many cases, a string that is not properly terminated with '\0' will result in illegal access of memory.



# 3.5 Importance of '\0' in a String (2/2)

What is the output of this code?

```
Without Null Char.c
#include <stdio.h>
#include <string.h>
                              One possible output:
                              Length = 8
int main(void) {
  char str[10];
                              str = apple¿ø<</pre>
                     Compare the output if you add:
  str[0] = 'a';
                     str[5] = ' \ 0';
  str[1] = 'p';
  str[2] = 'p';
                     or, you have:
  str[3] = '1';
                     char str[10] = "apple";
  str[4] = 'e';
  printf("Length = %d\n", strlen(str));
  printf("str = %s\n", str);
  return 0;
```

printf() will print %s from the starting address of str until it encounters the '\0' character.



%s and string functions work only on "true" strings. Without the terminating null character '\0', string functions will not work properly.

#### Quiz

 Please Arrays, Strings and Structures Quiz 1 before 3 pm on 23 August 2022.



CS2100 Arrays, Strings and Structures Quiz 1

Not available until 17 Aug at 0:00 | Due 23 Aug at 15:55



:



# **End of File**

