C: arrays; strings; input

COSC 208, Introduction to Computer Systems, 2021-09-06

Outline

- Warm-up: arrays
- Strings
- Input

Warm-up: arrays

- Q1: *What is the output of this program?*

```c
int main() {
    int sum = 0;
    int nums[] = { 1, 3, 5, 7 };
    for (int i = 0; i < 3; i++) {
        nums[i+1] -= 1;
        sum += nums[i];
    }
    printf("%d\n", sum);
}
```

```
7
```

- Q2: *What is the output of this program?*

```c
int main() {
    int sum = 0;
    int zeros[10];
    for (int i = 0; i < 10; i++) {
        sum += zeros[i];
    }
    printf("%d\n", sum);
}
```

Undefined — variables are not initialized like they are in Java and Python

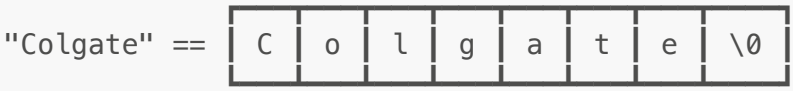
- Q3: *What is the output of this program?*

```c
int main() {
    int sum = 0;
    int nums[] = { 1, 2, 3 };
    for (int i = 0; i <= 3; i++) {
        sum += nums[i];
    }
    printf("%d\n", sum);
}
```

Undefined — C doesn't check array bounds like Java and Python

Strings

- String is simply an array of characters
- End of string is denoted by the null character (`\0`)



- Useful string functions
  - `strlen` counts the number of characters in an array before a null character
    - The null character is **not** included in the length
  - `strcmp` checks if the two strings are the same
    - Stops when it reaches a null character in either array
  - `strcpy` copies the characters from one array to another
    - Also copies the null character
  - The `man` pages for these functions indicate the parameters are are of type `const char *` or `char *`
    - `const` means the function does not modify the array
    - `char *` means a character pointer; in a few weeks we'll discuss the duality between arrays and pointers; for now, it means you can pass an array of characters to these functions

Input

- Use `fgets` to read in a line of input as a string

```c
char str[10];
fgets(str, 10, stdin);
```

  - `stdin` means *standard input*

Practice with strings and input

- Q4: *What is the output of this program?*

```c
int main() {
    char first[] = "Colgate";
    char second[10] = "Univ";
    printf("%lu %lu\n", strlen(first));
    printf("%lu %lu\n", strlen(second));
    first[strlen(first)] = '-';
    second[strlen(second)-1] = '.';
    printf("%s%s\n", first, second);
    first[3] = '.';
    first[4] = '\0';
    printf("%s %s\n", first, second);
}
```

```
7
4
Colgate-��Uni.
Col. Uni.
```

- Q5: *What is the output of this program?*

```c
int main() {
    char first[] = "Systems is fun!";
    char second[] = "Systems is fun!";
    if (first == second) {
        printf("1st == 2nd\n");
    }
    if (strcmp(first, second) == 0) {
        printf("1st cmp 2nd\n");
    }
    if (first == first) {
        printf("1st == 1st\n");
    }
    if (strcmp(first, first) == 0) {
        printf("1st cmp 1st\n");
    }
}
```

```
1st cmp 2nd
1st == 1st
1st cmp 1st
```

- Q6: *Write a program that asks the user for a string and prints the string backwards.*

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[100];
    printf("String? ");
    fgets(str, 100, stdin);
    for (int i = strlen(str); i >= 0; i--) {
        printf("%c", str[i]);
    }
    printf("\n");
}
```

Extra practice

- Q7: *Write a function called `avg` that takes an array of integers and the length of the array and returns the average of those integers.*

```
int avg(int nums[], int length) {
    int sum = 0;
    for (int i = 0; i < length; i++) {
        sum += nums[i];
    }
    return sum / length;
}
```

- Q8: *Write a function called `count` that takes an array of integers, the length of the array, and an integer to search for and returns the number of times the specified integer appears in the array.*

```
int count(int nums[], int length, int find) {
    int occurrences = 0;
    for (int i = 0; i < length; i++) {
        if (nums[i] == find) {
            occurrences++;
        }
    }
    return occurrences;
}
```

- Q9: *Write a program that asks the user for a string and converts all lowercase letters to uppercase and all uppercase letters to lowercase; numbers and punctuation should be left unchanged.*

```
#include <stdio.h>
#include <stdlib.h>
```

```c
#include <string.h>
int main() {
    char str[100];
    printf("String? ");
    fgets(str, 100, stdin);
    for (int i = 0; i < strlen(str); i++) {
        if (str[i] >= 'A' && str[i] <= 'Z') {
            str[i] = str[i] - 'A' + 'a';
    } else if (str[i] >= 'a' && str[i] <= 'z') {
            str[i] = str[i] - 'a' + 'A';
        }
    }
    printf("%s", str);
    return EXIT_SUCCESS;
}
```

- Q10: *Write a program that asks the user for a string and checks if the string is a palindrome (i.e., reads the same forwards and backwards).*

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char str[100];
    printf("String? ");
    fgets(str, 100, stdin);
    str[strlen(str)-1] = '\0'; // strip newline
    for (int i = 0; i < strlen(str) / 2; i++) {
        if (str[i] != str[strlen(str)-i-1]) {
            printf("Not a palindrome\n");
            return EXIT_SUCCESS;
        }
    }
    printf("Palindrome\n");
    return EXIT_SUCCESS;
}
```