C: arrays; strings; input; struct

COSC 208, Introduction to Computer Systems, 2022-02-01

Outline

- Warm-up I: arrays
- Strings & Input
- `structs`

Warm-up I: 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

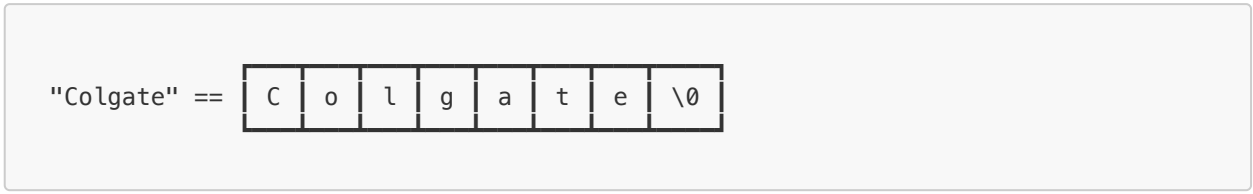
- Q3: *What is the output of this program?*

```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, i.e `\0`
  - 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

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

  - `stdin` means *standard input*

Practice with strings and input

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

```
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?*

```
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.*

```c
#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");
}
```

structs

- *How is a struct declared?*

```
struct tvshow {
    char name[100];
    int season;
};
```

- *How are fields of the struct accessed?* — with the dot (.) operator

```
struct tvshow favorite;
strncpy(favorite.name, "Tiny House Nation", 100);
favorite.season = 6;
```

- A struct is a *collection of values*
    - it is **not** an object, and hence
    - it cannot have methods associated with it
- A struct variable holds *values* for the **fields** of the struct;
    - it is **not** a reference to the struct, and hence
    - it is **copies** of the values that are passed to functions
- Q7: *What is the output of this program?*

```
struct one {
    char x;
    char y;
    short z;
};

struct two {
    int m;
    int n[10];
};

int main() {
    struct one a;
    struct two b;
    printf("%d %d\n", sizeof(struct one), sizeof(a.z));
    printf("%d %d\n", sizeof(b), sizeof(b.n));
}
```

```
4 2
44 40
```

- Q8: *What is the output of this program?*

```c
struct alpha {
    char x[10];
    int y;
};
struct beta {
    int b;
    int c;
};
int main() {
    struct alpha a = { "Colgate", 13 };
    struct beta b = { 1, 2 };
    struct beta c = { 3, 4 };
    a.y += -13;
    b.b = 5;
    c = b;
    b.c = 6;
    printf("a %s %d\n", a.x, a.y);
    printf("b %d %d\n", b.b, b.c);
    printf("c %d %d\n", c.b, c.c);
}
```

```
a Colgate 0
b 5 6
c 5 2
```

Array & string: extra practice

- QA: *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;
}
```

- QB: *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;
}
```

- QC: *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>
#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;
}
```

- QD: *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;
}
```

`struct`: extra practice

- QE: *Write a struct definition to represent a date (year, month number, and day).*

```c
struct date {
    int year;
    int month;
    int day;
};
```

- QF: *Write a function called `compare` that takes two date structs and returns -1 if the first date occurs before the second, 0 if the dates are equal, and 1 if the first date occurs after the second.*

```c
int compare(struct date a, struct date b) {
    if (a.year < b.year) {
        return -1;
    } else if (b.year < a.year) {
        return 1;
    } else {
        if (a.month < b.month) {
            return -1;
        } else if (b.month < a.month) {
            return 1;
        } else {
            if (a.day < b.day) {
                return -1;
            } else if (b.day < a.day) {
                return 1;
            } else {
                return 0;
            }
        }
    }
}
```