CS 340
Computer Systems

#3: C Programming and Logic Gates

Jan. 24, 2023 · Wade Fagen-Ulmschneider

Programming in C

Today, you'll begin your very first program in C!

- You already know how to program in C++! 🎉
- Programming in C is a simplification of the C++ programming.

1. Program Starting Point of **ALL** C PROGRAMS:

2. Printing Using **`printf()`** (and include **`<stdio.h>`**):

03/printf.c

```c
int main() {
  int i = 42;
  char *s = "Hello, world!";
  float f = 3.14;

  printf("%d  %s  %f\n", i, s, f);
  printf("%d\n", s[0]);
  printf("%d\n", s);
  printf("%d\n", f);
  return 0;
}
```

`printf` has a variable number of arguments:

<u>First argument</u>

<u>Additional arguments</u>

3. Pointers:

4. Heap Memory Allocation:

03/malloc.c

```c
typedef struct _myObject {
  int value;
  char *s;
} myObject;

int main() {
  char *s = malloc(10);
  myObject *obj = malloc(sizeof(myObject));
  obj->value = 3;

  printf("%p %p %d\n", s, obj, obj->value);
  return 0;
}
```

5. Strings – There is no "data type" in C known as a string. Instead, we refer to "C Strings" as a sequence of characters:

- A "C string" is just a character pointer: __________.
- The string continues until it reaches a __________ byte.
- C will automatically include the NULL byte **ONLY** when using double quotes in your code (***not*** *counted as part of the length, but* ***does require memory*** *–* ***<u>extremely tricky!</u>***)

03/string.c

```c
char *s = malloc(6);
strcpy(s, "cs340");
printf("s[0]: 0x%x == %d == %c\n", s[0], s[0], s[0]);
printf("s[4]: 0x%x == %d == %c\n", s[4], s[4], s[4]);
printf("s[5]: 0x%x == %d == %c\n", s[5], s[5], s[5]);
printf("s == \"%s\", strlen(s): %ld\n\n", s, strlen(s));

char *s2 = s + 2;
printf("s2[0]: 0x%x == %d == %c\n", s2[0], s2[0], s2[0]);
printf("s2 == \"%s\", strlen(s2): %ld\n\n", s2, strlen(s2));

*s2 = 0;
printf("s2[0]: 0x%x == %d == %c\n", s2[0], s2[0], s2[0]);
printf("s2 == \"%s\", strlen(s2): %ld\n\n", s2, strlen(s2));

printf("s == \"%s\", strlen(s): %ld\n", s, strlen(s));
```

...what is happening in memory?

03/utf8.c

```c
char *s = malloc(5);
s[0]=0xF0; s[1]=0x9F; s[2]=0x8E; s[3]=0x89; s[4]=0x00;

char *s1 = "\xF0\x9F\x8E\x89";
char *s2 = "🎉";
char *s3 = "\U0001f389";    // \U - must be 8 bytes

printf("%s %s %s %s\n", s, s1, s2, s3);
printf("strlen(): %ld %ld %ld %ld\n", strlen(s), strlen(s1),
                                      strlen(s2), strlen(s3));
```

...how can we represent non-ASCII characters in C code?

Some extremely useful built in string functions:

- **`strcmp(char *s1, char *s2)`** -- Compares two strings
- **`strcat(char *dest, char *src)`** -- Concatenate two strings
- **`strcpy(char *dest, char *src)`** -- Copies a string
- **`strlen(char *s)`** -- Returns the length of the string

Logic Gates and Truth Tables

We can begin to define the building blocks of the CPU by basic instructions with input bits and output bits through **logical gates**.

- By convention, you will see that the input bits are labeled **`A`** and **`B`** by default.

Logic Gate #1:

Logic Gate #2:

Logic Gate #3:

Logic Gate Challenge: ***`A XOR B`***

We can also express this in a table known as a **truth table**:

Op.	Binary	Math	Example Values		
	A	x	1100	110011	101
	B	y	1010	11	010
AND	A & B	xy			
OR	A | B	x + y			
XOR	A ^ B	x XOR y			
NOT	!A	x'			

Truth Table: Half Adder

A	B	A + B	SUM	CARRY

Truth Table for a Half Adder

Circuit Diagram for a "Half Adder":

Full Adder:

A	B	$CARRY_{in}$					SUM	$CARRY_{out}$

Truth Table for a Full Adder

Circuit Diagram for a "Full Adder":

Chaining Circuits Together: ______________________________

Disadvantages: