Lecture 3:
Binary Operators and Strings

CSE 29: Systems Programming and Software Tools

Aaron Schulman (Shalev)

Lecture 3 Overview

- Examples of processing strings in C
- How we can control individual bits in memory in C

Week 1 Announcements

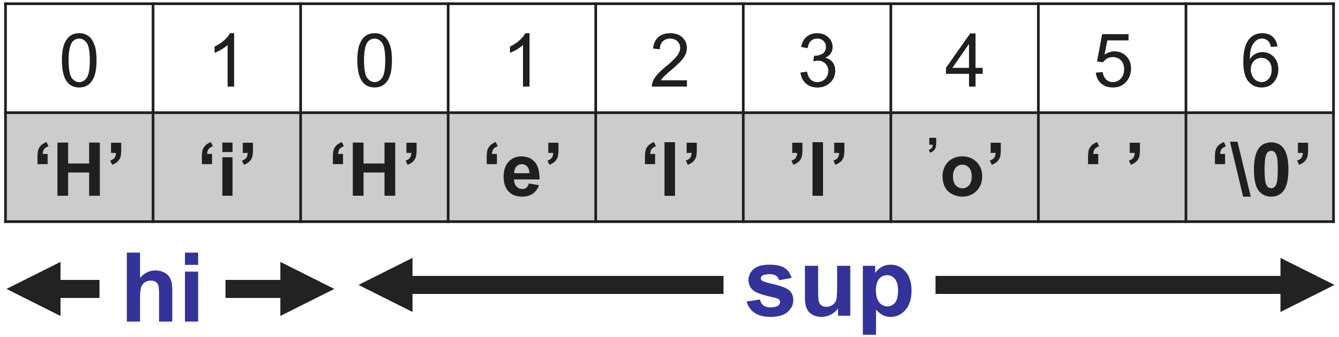
- **Homework 1 is available on PrarieLearn**
  - Get started immediately and go to all available office hours
  - Start all parts that you can do right now (ASCII+strings)
  - Due on Monday the 6th
    - But don’t worry, multiple submission attempts!

Demo: How do we compute string length?

- **Big Idea:**
  - There are no training wheels anymore in C, this is not Java
  - If you tell the computer to do something, it will do exactly what you say.
  - Look for zeros, it will look for zeros

What happens if the null is not there?

```
char sup[7] = “Hello”;
char hi[2] = {‘H’, ‘i’};
```



0	1	0	1	2	3	4	5	6
‘H’	‘i’	‘H’	‘e’	‘l’	’l’	’o’	‘ ’	‘\0’

All variables share the same linear memory space

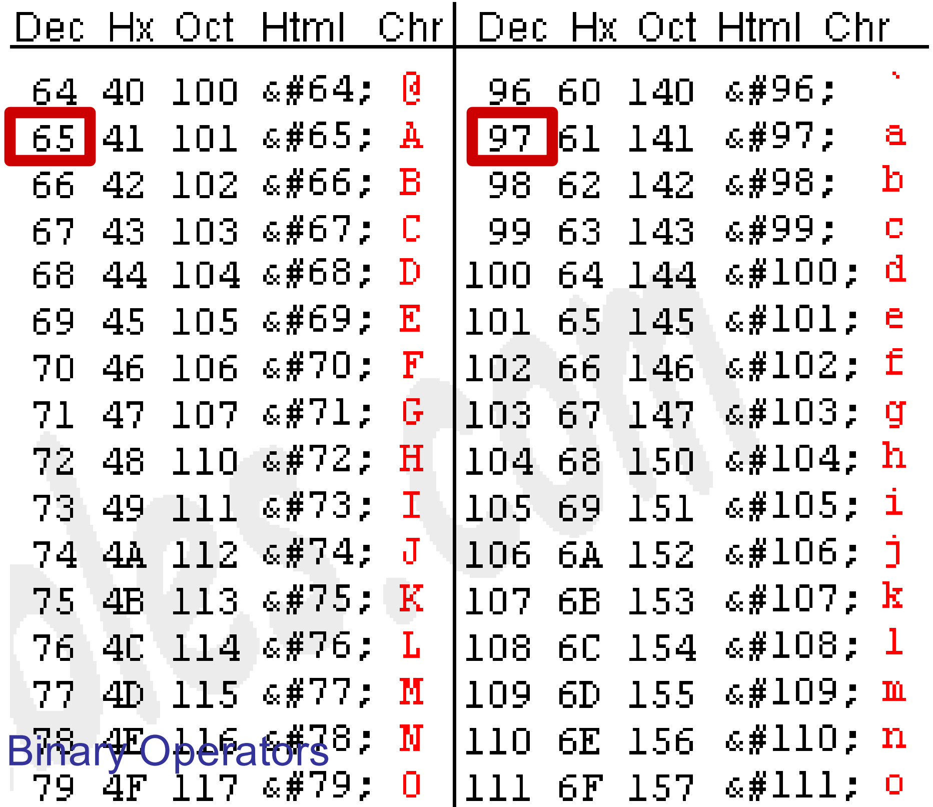
The “char” data type (one byte also int8_t)

- Storing human readable English characters
  - char ch = ‘A’; // 1 byte, single quote needed

- ASCII: The English characters have number equivalents

A = 65	a= 97	0 = 48	! = 33
B= 66	b = 98	1 = 49	“ = 34
...	...	...	...

How can we go from Upper to Lowercase?



Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	&#64;	@	96	60	140	&#96;	`
65	41	101	&#65;	A	97	61	141	&#97;	a
66	42	102	&#66;	B	98	62	142	&#98;	b
67	43	103	&#67;	C	99	63	143	&#99;	c
68	44	104	&#68;	D	100	64	144	&#100;	d
69	45	105	&#69;	E	101	65	145	&#101;	e
70	46	106	&#70;	F	102	66	146	&#102;	f
71	47	107	&#71;	G	103	67	147	&#103;	g
72	48	110	&#72;	H	104	68	150	&#104;	h
73	49	111	&#73;	I	105	69	151	&#105;	i
74	4A	112	&#74;	J	106	6A	152	&#106;	j
75	4B	113	&#75;	K	107	6B	153	&#107;	k
76	4C	114	&#76;	L	108	6C	154	&#108;	l
77	4D	115	&#77;	M	109	6D	155	&#109;	m
78	4E	116	&#78;	N	110	6E	156	&#110;	n
79	4F	117	&#79;	O	111	6F	157	&#111;	o

Demo: We can just add 32, right?

- **Big Idea:**
  - Addition can indeed change just one bit from 0 to 1
  - But, it also will add 1 to a bit that is already 1

Accessing individual bits in C data types

It is possible to control/access individual bits of Integer data types in C

- int*_t datatypes all can have bit-wise operations

There are special mathematical operators in C for doing operations on bits

- & - AND
- | - OR
- ^ - XOR
- >> - Shift Right
- << - Shift Left

Boolean operators work on all bits of a variable (e.g., 8 bits in a char)

How bit operations work: **OR**

```
char a = 2; // 00000010 in binary
char b = 5; // 00000101 in binary

// OR each bit of the two integers together
char a_or_b = a | b;
//   00000010
// | 00000101
// -----------
//   00000111

printf(“%d\n”, a_or_b); // What will the output be?
```

How bit operations work: **AND**

```
char a = 3; // 00000011
char b = 5; // 00000101

// AND each bit of the two integers together
char a_and_b = a & b;
//   00000011
// & 00000101
// -----------
//   00000001 // Lets you select bits you want to inspect

printf(“%d\n”, a_and_b); // What will the output be?
```