

Announcements

Office Hours

Today: 1:00—3:00

Monday, Wednesday: 1:00—3:00

Tuesday: 2:30-4:00 (adjustments may be needed)

Lab partners: We are in the middle of stacks and queues,
New Partners the Wednesday after Thanksgiving

Guest instructor Wednesday (maybe Friday also)

Important to come to Wednesday's class on time and prepared!!!!

All work next week to be submitted to my office, NOT IN CLASS

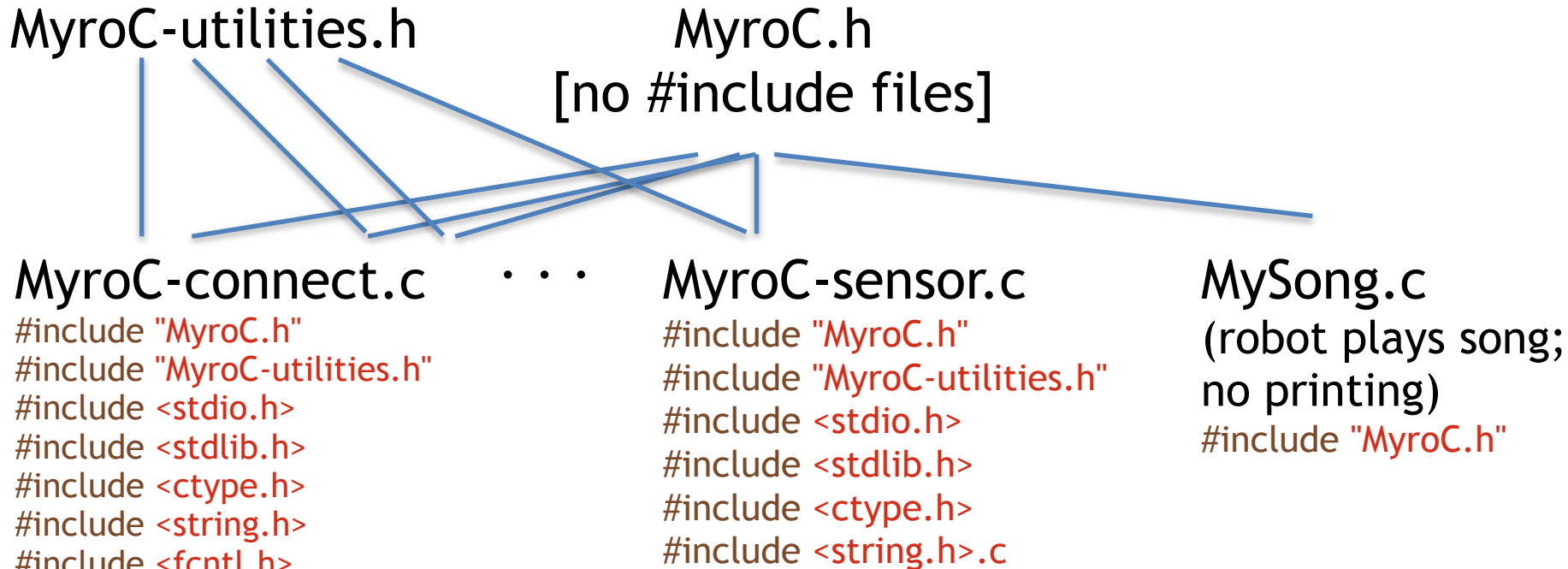
**Work turned in during class on Wednesday, Friday will be
assigned –10 (negative 10) points and shredded**

"Thursday" Extras from Tenure-track Candidates

- Tuesday @ 4:00: William Bares, College of Charleston; intelligent interactive graphical systems
- Thursday @ 4:00: Yucong Jiang, Indiana University; music informatics

Notes about Program Management

Consider the MyroC package, its structure, and its use



implementation divided into several files,
compiled into a single object file

Key Notes:

- MyroC.h contains no #include statements
- MyroC-utilities.h contains only 1 #include—needed for function signatures
- Each C implementation file only includes what it needs

Announcements

Looking ahead to the last week of classes

- only 2 extra credit supplemental problems may be submitted during the last week of classes 5:00 pm Friday, 7 December, through 5:00 pm Friday, 14 December 2019
- No work accepted after 5:00 pm Friday, 14 December 2019
- If you have a partner preference for the last week, email me your preference by noon, Sunday, December 8. (To be honored, both partners need to email me.)
- I'll assign partners for last week, if no preferences given.

Queues

Uses:

scheduling work inside a computer

printing

simulations (e.g., traffic at a stop light, customers in store)

many advanced algorithms

Logical interface

Common implementations

Arrays: several interpretations

Lists: Monday

Queue user interface

Data type (for storing strings on a queue)

```
typedef . . . {  
    } stringQueue;
```

Operations:

Vote 1: Which header is appropriate for an initializeQueue operation?

- A. void initializeQueue (stringQueue queue)
- B. void initializeQueue (stringQueue * queue)
- C. Either A or B
- D. Something else?

Vote 2: Which header is appropriate for an isEmpty operation?

- A. int isEmpty (stringQueue queue)
- B. int isEmpty (stringQueue * queue)
- C. Either A or B (no preference)
- D. A preferred (but B would work)
- E. B preferred (but A would work)

Queue user interface

Data type (for storing strings on a queue)

```
typedef . . . {  
    } stringQueue;
```

Operations:

```
/* initialize queue referenced by the parameter */  
void initializeQueue (stringQueue * queue)  
  
/* return whether the queue is empty */  
int isEmpty (stringQueue queue)  
  
/* return whether the queue is full */  
int isFull (stringQueue queue)  
  
/* insert given item into back of the queue */  
int enqueue (stringQueue * queue, char* item)  
  
/* removes front item from queue referenced by parameter */  
char * dequeue (stringQueue * queue)
```

Array Implementation of a Queue

Elements of a queue (e.g., a, b, c, d, e) are stored in an array

Idea 1: head of list at position 0, end of list at variable last

Index	0	1	2	3	4		
	a	b	c	d	e		
notes	front				last		

On dequeue, slide elements left:

Index	0	1	2	3	4		
	b	c	d	e			
notes	front			last			

This approach is:

- A. both enqueue, dequeue efficient
- B. enqueue efficient; dequeue not
- C. dequeue efficient; enqueue not
- D. neither efficient

Array Implementation of a Queue

Elements of a queue (e.g., a, b, c, d, e) are stored in an array

Idea 2: variables head, tail keep track of ends of queue

Index	0	1	2	3	4		
	a	b	c	d	e		
notes	head				tail		

On dequeue, remaining elements not moved:

Index	0	1	2	3	4		
		b	c	d	e		
notes		head			tail		

This approach is:

- A. both enqueue, dequeue efficient
- B. enqueue efficient; dequeue not
- C. dequeue efficient; enqueue not
- D. neither efficient

Array Implementation of a Queue

Elements of a queue (e.g., a, b, c, d, e) are stored in an array

Idea 2: variables head, tail keep track of ends of queue

Index	0	1	2	3	4		
	a	b	c	d	e		
notes	head				tail		

State after several enqueues and dequeues:

Index	0	1	2	3	4		
			c	d	e	f	g
notes			head				tail

What should happen when tail gets to end of array

- A. declare queue full, and accept no more elements
- B. slide elements, so head at 0
- C. restart tail with index 0; when head eventually gets to end, it also restarts at 0 (indices wrap around as elements enter and leave queue)

Array Implementation variations

Index	0	1	2	3	4	5	6
		b	c	d	e	f	

Possible interpretations of head, tail

1. tail indicates index where item just added (e.g., 5)
2. tail indicates index where next item will be added (e.g., 6)

Initialization of queue variables, assuming first element will be placed in array at index 0

- Option 1: Initialize tail=6 for interpretation 1
- Option 2: Initialize tail=0 for interpretation 1
- Option 3: Initialize tail=6 for interpretation 2
- Option 4: Initialize tail=0 for interpretation 2

Which options would work properly?

A. Options 1, 3

C. Options 2, 3

E. Something else

B. Options 1, 4

D. Options 2, 4

Array Implementation variations

Index	0	1	2	3	4	5	6
		b	c	d	e	f	

Possible interpretations of head, tail

1. head indicates index where next element will be removed (e.g., 6)
2. head indicates index where front item just removed (e.g., 0)

Initialization of queue variables, assuming first element will be placed in array at index 0

- Option 1: Initialize head=6 for interpretation 1
- Option 2: Initialize head=0 for interpretation 1
- Option 3: Initialize head=6 for interpretation 2
- Option 4: Initialize head=0 for interpretation 2

Which options would work properly?

- A. Options 1, 3 C. Options 2, 3 E. Something else
- B. Options 1, 4 D. Options 2, 4