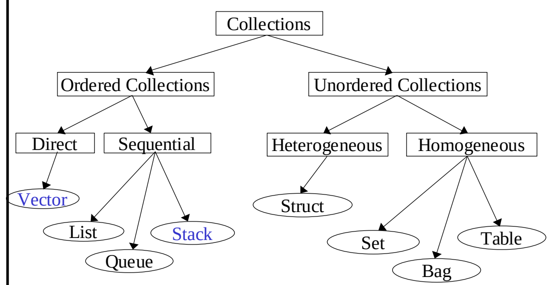
CSE 143

Queues, Event Lists, and Simulations

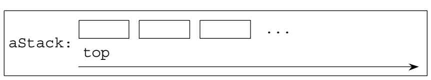
Chapter 7

A Classification of ADTs



A Stack Definition (Review)

- Stack: "Homogeneous, ordered collection, accessed only at the front for removal and insertion"
  - *Top: last value in, first value out*

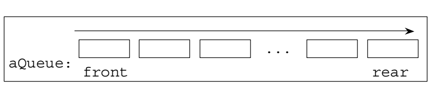


When A Stack Is Not Quite Right

- People line up to use a pay phone
- There's only one place you leave the line (at the front, when the phone is free)
- There's only one place you enter the line (at the end -- everyone assumed to be polite!)
- The two places are different! So it's not a stack
- First in, first out: a "queue"

Queue Definition

- Queue: "Homogeneous, ordered collection, accessed only at the front (removal) and rear (insertion)"
  - *Front*: First element in queue
  - *Rear*: Last element of queue
- *FIFO*: First In, First Out



Abstract Queue Operations

- *insert(item)* : Adds an element to rear of queue
  - succeeds unless the queue is full
  - often called "enqueue"
- *item remove( )* : Removes and returns element at the front of queue
  - succeeds unless the queue is empty
  - often called "dequeue"
- *item getFront( )* : Returns a copy of the front element of queue
  - succeeds unless the queue is empty
- No cursor, no iteration

Queue Example

- Draw a picture and show the changes to the queue in the following example:

```
Queue q; int v1, v2;

q.insert(4);
q.insert(7);
q.insert(5);
v1 = q.remove();
v2 = q.getFront();
q.insert(9);
q.insert(2);
```

3/25/2001 P-7

What is the result of:

```
Queue q; int v1,v2,v3,v4,v5,v6
q.insert(1);
q.insert(2);
q.insert(3);
v1 = q.remove();
v2 = q.getFront();
q.insert(4);
v3 = q.remove();
v4 = q.getFront();
q.insert(5);
v5 = q.remove();
v6 = q.getFront();
```

3/25/2001 P-8

Queue vs. Stack

If I put a bunch of stuff in a queue or stack and then take it all out again...

- Queue: get it back in the original order
- Stack: get it back in reverse order

- Does this suggest an algorithm for checking palindromes??

3/25/2001 P-9

Common Uses of Queues

- Internal to a computer
  - Processes in an operating system, waiting for service (CPU, I/O, etc.)
  - Buffering input/output
    - When printing to screen with cout, characters don't appear right away: they are buffered and sent out in groups.
    - Print jobs at the printer
  - Mouse events needed to be handled
- Simulations
  - People or things waiting in line for service
  - More about this later

3/25/2001 P-10

Public Queue Interface

```
class IntQueue {  //see p.310
public:
  IntQueue( );          //should define a copy constructor, too
  bool isEmpty( );
  void insert(int item);
  int remove( );
  int getFront( );
private:
  // more than one way to do it.  Should the client care?
};
```

3/25/2001 P-11

Queue Implementations

- Choices similar to Stack
  - Array-based
  - Linked list
  - Existing Vector ADT implementation
- Queues are a little more complicated than Stacks
  - More complexity in the implementation
  - More ways to do it

3/25/2001 P-12

Queue Implementations

- Choices similar to Stack
  - Array-based
  - Linked list
  - Use existing Vector ADT

```
Convention: Rear is last position of vector
Convention: Front is always position 0
Quite straightforward, very few lines of code
```

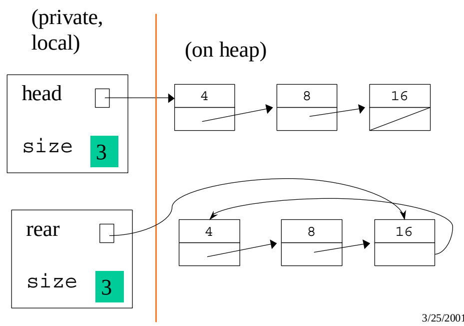
3/25/2001 P-13

Another Implementation

- Choices similar to Stack
  - Array-based
  - Linked list
  - Exisiting Vector ADT implementation
- With L.L. we could follow Vector ADT pattern
  - Insert at end of linked list
  - Remove from front of list
  - A separate “rear” pointer would increase efficiency
- Another linked-list approach: a circular list
  - Let rear node point back to head node

3/25/2001 P-14

Conventional vs. Circular



3/25/2001 P-15

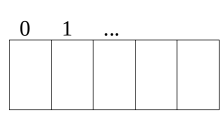
Array Implementation

- Choices for implementation
  - Array-based
  - Linked list
  - Exisiting Vector ADT implementation
- Stack: nItems was enough
- Queue: need to keep track of front and rear separately
  - still need to recognize empty queue, full queue

3/25/2001 P-16

First Attempt

- int front, rear;
- Start at index 0, add new items left to right
- Insert: rear++
- Delete: front ++

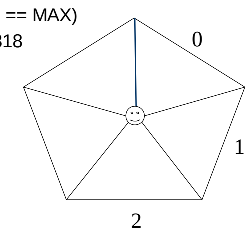


This is a good start, but over time... what goes wrong?

3/25/2001 P-17

Wrap Around

- int size, front, rear;
- rear = (rear+1) % MAX; //on enqueue
- queueEmpty = (size == 0)
- queueFull = (size == MAX)
- Textbook p.317-318



3/25/2001 P-18

Computers and Simulation

- Computer programs are often used to “simulate” some aspect of the real world
  - Movement of people and things
  - Economic trends
  - Weather forecasting
  - Physical, chemical, industrial processes
- Why?
  - Cheaper, safer, more humane
  - But how accurate?

3/25/2001 P-19

Simulation of a Bank

- People arrive and get in line for a teller
- When a teller is free, person at the head of the line gets served
  - Sounds like a queue is the right data model
- A bank might have different kinds of “tellers” (commercial tellers, loan officers, etc)
  - different queues for each one
- See textbook pp.324-334

3/25/2001 P-20

Queues and Simulations

- Queues are often useful in simulations
  - because queues occur in the real world
- Common considerations
  - Time between arrival
  - Service time
  - Number of servers
- Often want to investigate/predict
  - Time spend waiting in queue
  - Effect of more/fewer servers
  - Effect of different arrival rates

3/25/2001 P-21

Simulation vs Calculation

- Business question: *should the bank hire an additional teller?*
- Take a simplified situation:
  - People arrive on average one per minute
  - Each person spends on average one minute with the teller
  - Might be passed on historical statistics
- By calculation: each person waits on average one minute

Would simulation give a different picture?

3/25/2001 P-22

Airport 2000

- Planes approach the airport
- Wait for a controller
- Are assigned one of three runways
- Land and clear the runway
- What happens with:
  - more/fewer/differently scheduled planes?
  - more/fewer controllers?
  - another runway?
  - etc.
- What would computer model look like?

3/25/2001 P-23

Simulations in Science

- Classical physics: describe the physical world with (differential) equations
  - Problem: too many interactions, equations too numerous and complex to solve exactly
- Alternative: build a model to simulate the operation
- Zillions of applications in physics, weather, astronomy, chemistry, biology, ecology, economics, etc. etc.
- Ideal model would allow safe virtual experiments and dependable conclusions

3/25/2001 P-24

Time-Based vs. Event Based

- Time-based simulation
  - Look and see what happens at every "tick" of the clock
- Might "throw dice" to determine what happens
  - Random number or probability distribution
- Size of time step?
  - A day, a millesecond, etc. depending on application
  - Simulating the Big Bang vs formation of Grand Canyon
  - Are smaller steps better than larger steps?

3/25/2001 P-25

Time-Based vs. Event Based

- Event-based simulation
  - Schedule future events and process each event as its time arrives
- Bank simulation events
  - "Customer arrives" could be one event (external)
  - "Customer starts getting service" (internal)
  - "Customer finishes transaction"
  - "Teller goes to lunch"...
- Event list holds the events waiting to happen
  - Each one is processed in chronological order
  - External events might come from a file, user input, etc.
  - Internal events are generated by other events

3/25/2001 P-26

Grand Canyon Simulation

- Model: erosion due to amount and angle of water flow, type of material in top layer, etc.
- Events
  - Rain shower every 1-20 days (depending on season), flood event after 4 days of rain, earthquake every 40 years (followed by aftershock events), etc.
- Initial event list:

3/25/2001 P-27

EventList ADT

- See textbook p.334
- Holds events with their arrival times
- Operations:
  - EventListIsEmpty
  - EventListInsert
    `//based on time that event will happen`
  - EventListDelete
    `//process current (first) event`
  - EventListRetrieve
    `//see next event without deleting`

3/25/2001 P-28

Summary

- Stack
  - List with LIFO structure
  - Access via push(item), pop(), and top()
- Queue
  - List with FIFO structure
  - Access via insert(item), delete(), and getFront()
- Simulation
  - Computer model of a dynamic real-world situation
- EventList

3/25/2001 P-29

Looking Ahead

- Vector, Queue, Stack, EventList, etc. seem to be related
- One view of the relationship
  - Queue and Stack might <u>contain</u> a Vector as part of their implementation
- A different view
  - Queue and Stack might be a <u>kind of</u> a Vector
    `Similar data and operations, but not identical`
    `Maybe need the old data, but some new data`
    `Maybe could reuse some old operations, but have to modify some or add new operations`
  - Likewise, maybe EventList is a <u>kind of</u> a Queue

3/25/2001 P-30

Where Are We In The Course?

- One multiple-part project coming up: simulation
  - Due in stages; final part open-ended
- Lots of other course content yet to come
  - Algorithm efficiency
  - Sorting and searching
  - Trees and other data structures
- Much of this content will not be covered in any programming assignment
  - Will be on quizzes, exams, etc.
  - Will show up in future courses, programming projects, etc.