

Queues

Most important things to know about queues:

- First in, First out (FIFO)
- They are also a *Collection ADT*, in that they maintain a collection of items.
- They can be implemented with any type of ordered list object, which right now includes lists, arrays, and vectors (stretchy arrays).
- They have many uses! (Can you think of a few?)

Queue example: the PRINT QUEUE

When you print something on a computer, the print job goes into a queue. The last job to be submitted to the queue is the last job printed.

Using linked lists as our collection object, let's implement a print queue:

There are two main methods to Queues. *Insert*, which inserts something at the end of the queue, and *Remove*, which takes off the lead element and returns it.

In psuedo-code, here's what we'll have to do for each of these:

Insert (string filename):

- Make a new node with the correct filename.
- Insert this node at the tail of the list.

Special case: What if the list is empty? Do we need to do anything extra to maintain the properties of the queue?

Remove (??):

- Get a pointer to the last element.
- Make a copy of the last element.
- Return the last element.

Special case: If you maintain a pointer to the end of the list, don't forget to update it!

Now, lets look at the real thing. . .

Queues, part 2

```
// class spec for Node class
class Node {
    public:
        Node(string name) { jobName = name; next = NULL; }
        void print() { cout << jobName << endl; }
        string jobName; Node* next;
};
```

```
// class spec for PrintQueue class
#include "Node.h"

class PrintQueue {
    public:
        PrintQueue();
        ~PrintQueue();
        void insert(string);
        Node remove();
    private:
        Node* head;
        Node* tail;
};
```

```
// implementation of the PrintQueue class
#include "PrintQueue.h"

PrintQueue::PrintQueue() { head = tail = NULL; }

void PrintQueue::insert(string name) {
    if (tail == NULL) {
        tail = new Node(name);
        head = tail; }
    else {
        tail->next = new Node(name);
        tail = tail->next; }
}

Node PrintQueue::remove() {
    Node temp1 = *head;
    Node* temp2 = head;
    head = head->next;
    delete temp2;
    return temp1;
}

PrintQueue::~PrintQueue() {
    Node* temp;
    while (head != NULL) {
        temp = head->next;
        delete head;
        head = temp; }
}
```

Queues, part 3

Now, given the following main program, what is outputted to the screen?

```
#include "PrintQueue.h"

int main () {

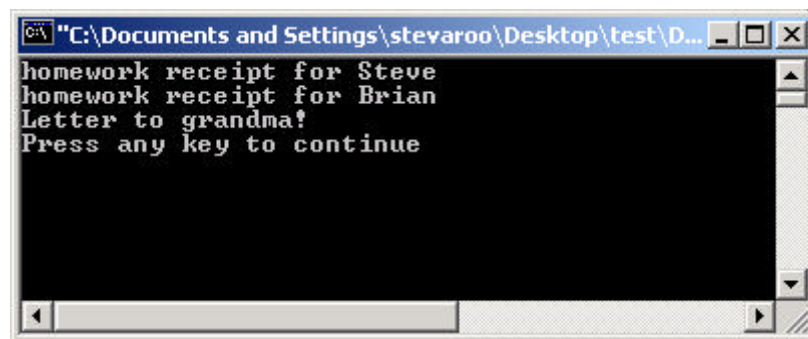
    PrintQueue MsWord;

    MsWord.insert("homework receipt for Steve");
    MsWord.insert("homework receipt for Brian");
    MsWord.insert("Letter to grandma!");

    (MsWord.remove()).print();
    (MsWord.remove()).print();
    (MsWord.remove()).print();

    return 0;
}
```

The answer is below, straight out of MSVC++:

A screenshot of a Windows command prompt window. The title bar shows the path "C:\Documents and Settings\stevaroo\Desktop\test\D...". The window contains the following text: "homework receipt for Steve", "homework receipt for Brian", "Letter to grandma!", and "Press any key to continue". The text is displayed in a monospaced font on a black background.

```
"C:\Documents and Settings\stevaroo\Desktop\test\D...
homework receipt for Steve
homework receipt for Brian
Letter to grandma!
Press any key to continue
```

Note the order that the results are removed from the queue!!

Is this a LIFO or a FIFO ADT? How do we know?