CS162 Lecture 20

Iterator Details
Housekeeping - Exam 2 and Final

- Exam 2 is Monday, Nov.19th!
- No class Wednesday Nov.21 (Happy Thanksgiving!)
- No recitation that week either
- Friday is Exam 2 practice questions
- Today is the detailed review of Iterators that we didn’t have time for last week.

- Final is a take-home test, released Wed Nov.28, due as PDF Wed Dec.5
- Totally different from exams 1 and 2, fugehttaboudit until after break.
- Assignment 4 is posted online!
- We have already covered everything you need for it!
- Design document due **Sunday 11/18**, Assignment due **Sunday 11/25**!

- Turn in assignment 4 by 11/18 for 5 pts extra credit!
- Demo assignment 4 by 11/25 for 5 pts whether you had it in by 11/18 or not!
- Must have assignment 3 Demo’d to qualify on either count
- **TA’s opening office hours for drop-in demos**
Exam 2

- Same format as Exam 1
- True/False, Multiple Choice only
- In-class
- 50 minutes
- On Scantron
- All material is fair game, but heavily weighted to material since exam 1
- Makes sense, as new material requires you use old material anyway
- Example: Big 3 for template classes?
  - Requires you understand big 3, even though question is about templates
Iterators

- A construct that allows you to cycle through the data items stored in a data structure
- Generalization of a pointer: typically implemented as a pointer but is abstracted so we don’t have to deal with it
- Every container class in STL has its own iterator
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> v;
    for(int i=0; i<5; i++) {
        v.push_back(i);
    }
    cout << "Print v" << endl;
    vector<int>::iterator itr;
    for(itr = v.begin(); itr != v.end(); itr++) {
        cout << *itr << " ";
    }
    cout << endl;
    return 0;
}
Commonly Overloaded Operators for Iterators

++ move to next item

-- move to previous item

== equality test

!= not equal test

* dereference operator
Types of Iterators

- Forward: ++ works on iterator
- Bidirectional: both ++ and -- work on the iterator
- Random Access: ++, --, and random access all work
- Constant: doesn’t allow changes to be made to element at its location
- Mutable: can change the element at its location
- Reverse: can be used to cycle through all elements of a container with bidirectional iterators
- Input: forward iterator that can be used with input stream
- Output: forward iterator to be used with output stream
Linked Lists

- A list constructed using pointers
- Can grow and shrink while the program is running
- Not stored contiguously in memory
- Use nodes (struct) to create
- **Ptr to first node is typically called “head”**

```c
struct Node {
    int val;
    node* next;
}
```
procedure bubbleSort( A : list of sortable items )
    n = length(A)
    repeat
        swapped = false
        for i = 1 to n-1 inclusive do
            /* if this pair is out of order */
            if A[i-1] > A[i] then
                /* swap them and remember something changed */
                swap( A[i-1], A[i] )
                swapped = true
            end if
        end for
    until not swapped
end procedure
What else?

- Final will be take-home, different format from exams 1 and 2
- You’ll be analyzing the runtime complexity of searching algorithms
- We will cover this material after you’re back from Thanksgiving Break
- We will also be covering what you need for Assignment 5 after you’re back from Thanksgiving Break
- All of this to say, the only thing you will have on your minds over Thanksgiving is Assignment 4
- Ideally you’d have nothing to do over break, but there’s only so much I can do
- The good news is you will have everything you need for the assignment, no exam to study for, and no reason to worry about an “all or nothing” final
- I can’t make this class “not difficult”, but I hope this arrangement helps to make it “not stressful”