Course goals

- exposure to another language
  - C++
  - Object-oriented principles
- knowledge of specific data structures
  - lists, stacks & queues, priority queues, dynamic dictionaries, graphs
- impact of DS design & implementation on program performance
  - asymptotic complexity of algorithms

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (insertion sort vs mergesort)
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - Traversal
  - Shortest paths

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (i
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - Traversal
  - Shortest paths

- objects
- classes -- .h and .cpp files
- templates
- access control
  - public/ private/ protected methods
  - friend classes
- inheritance
  - public/ private/ protected inheritance
  - virtual functions
  - abstract classes

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

Review of C++

Introduction to Unix

Review of program performance
- time and space complexity
- asymptotic notation

-- *searching* (linear vs binary) & *sorting* (i

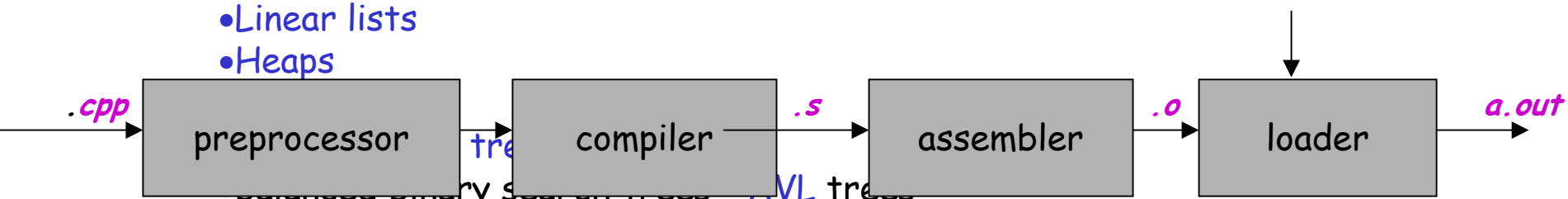
Data representation and lists

Stacks and Queues

Binary trees
- representation
- traversal

Priority queues
- Linear lists
- Heaps

- emacs? pico
- the g++ compiler
  - stages in compilation
- makefiles
- the gdb debugger
- man pages



Graphs
- representation
- Traversal
- Shortest paths

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (insertion sort vs mergesort)
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search tr
- Graphs
  - representation
  - Traversal
  - Shortest paths

- bigOh/ bigTheta notation
- asymptotic **worst-case** complexity of algorithms
- common complexities:
  - log n
  - n
  - n log n
  - n^2, n^3, ...
- determining complexities of algorithms
  - inspection
  - recurrences
- example complexities -- sort/ search

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sort*
- Data representation and lists
- Stacks and Queues
- Hash tables

- data representation:
  - array-based
  - linked/ pointer-based
- lists
  - ADT specification
  - representation using arrays
  - representation using linked lists
  - compare and contrast

```
class list{ //implementation in C++
  public:
     list();
     ~list();
     bool isEmpty();
     bool isFull();
     int length();
     bool Find(x,k);
     int Search(x);
     void delete(k,x);
     void insert(k,x);
  private:
};
```

```
adt linearList{
     create()
     destroy()
     isEmpty()
     isFull()
     length()
     Find(x,k)
     Search(x)
     delete(k,x)
     insert(k,x)
}
```

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sort*
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - Traversal
  - Shortest paths

- data representation:
  - array-based
  - linked/ pointer-based
- lists
  - ADT specification
  - representation using arrays
  - representation using linked lists
  - compare and contrast

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sort*
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - Traversal
  - Shortest paths

- ADT specification
  - stack - LIFO
  - queue - FIFO
- implementation
  - representation using arrays
    - "circular" for queues
  - representation using linked lists
  - $\Theta(1)$ time operations
  - min and max operations

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (insertion sort vs mergesort)
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - Traversal
  - Shortest paths

- a **recursive** definition
  - root
  - left [sub]tree
  - right [sub]tree
- implementation
  - representation using arrays
    - inefficient, except for complete trees
  - representation using linked structures
  - O(h) time operations (h: height of the tree)
- tree **traversals** -- recursively defined
  - preorder/ inorder/ postorder
  - each takes O(n) time (n: # elements)

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (insertion sort vs mergesort)
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search tre
- Graphs
  - representation
  - Traversal
  - Shortest paths

- ADT specification
  - create/ destroy/ isEmpty
  - insert
  - min
  - deleteMin
- implementation
  - linear list -- one of the operations is O(n)
  - binary tree -- a *complete* tree
    - represented using array
      - O(log n) operations
      - fast implementations (bit-manipulation)
- other operations --
  - max
  - decrease/ increase
  - delete

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment

of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performa
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binar
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - Traversal
  - Shortest paths

- dynamic dictionaries -- ADT
  - create/ destroy
  - insert
  - delete
  - find
- implementation using binary trees
  - bst's --operations are O(h)
    - inorder traversal sorts the elements
  - balanced bst's -- the AVL tree
    - height is always O(log n)
    - insert/ delete may involve rotations
      - RR/ LL/ RL/ LR

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *s*
- Data representation and lists
- Stacks and Queues
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - Traversal
  - Shortest paths

- definition: G=(V,E), |V| = n; |E|=m;
  - lots of terminology
- representation
  - adjacency matrices
  - adjacency lists
  - **compare** and **contrast**
- example operations
  - topological sort of DAG's
  - cycle detection
    - directed and undirected graphs
  - shortest paths
    - the Warshall-Floyd algorithm