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
Hash tables
Binary trees
  • representation
  • traversal
Priority queues
  • Linear lists
  • Heaps
Search trees
  • Binary search trees
  • balanced binary search trees – AVL trees
Graphs
  • representation
  • traversal
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
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees - AVL trees
- Graphs
  - representation
  - traversal

- objects
- classes -- .h and .cpp files
- templates
- access control
  - public/ private/ protected methods
  - friend classes
- inheritance
  - public/ private/ protected inheritance
  - multiple inheritance
- the strings package
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
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
- Graphs
  - representation
  - traversal

Program performance:
- time and space complexity
- asymptotic notation
- searching (linear vs. binary)
- sorting (insertion sort vs. mergesort)

Compilation stages:
- Preprocessor
- Compiler
- Assembler
- Loader

- .cpp
- .s
- .o
- a.out

- man pages
- the g++ compiler
- stages in compilation
- makefiles
- environment variables
- the gdb debugger
- emacs? pico
## 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
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees
- Graphs
  - representation
  - traversal

- 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
- example complexities -- sort / search
Example: merge sort

```plaintext
mergeSort(A, i, j) // sort A[i,...j]
{
    if (i==j) return A[];
    mergeSort(A, i, (i+j)/2);
    mergeSort(A, (i+j)/2 + 1, j);
    merge(A, i, (i+j)/2, j)
}

merge(A, i, k, j)
//PreCond: A[i,...,k] and A[k,...,j] are sorted
//PostCond:A[i,...,j] is sorted
```

Recurrence: \[ T(n) \leq 2 \cdot T(n/2) + c_1 \cdot n + c_2 \]
\[ T(1) = c_3 \]
\[ T(n) = O(n \log n) \]
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
Hash tables

class list{
public:
  list();
  ~list();
  bool isEmpty();
  bool isFull();
  int length();
  bool Find(x,k);
  int Search(x);
  void delete(k,x);
  void insert(k,x);
};

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) & sorting
- Data representation and lists
  - stacks and queues
  - hash tables
  - binary trees
    - representation
    - traversal
  - priority queues
    - linear lists
    - heaps
- Search trees
  - binary search trees
  - balanced binary search trees - AVL trees
- Graphs
  - representation
  - traversal

- data representation:
  - array-based
  - linked/pointer-based
  - simulated pointer (cursors)
- 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) & sorting
Data representation and lists
  Stacks and Queues
Hash tables
  Binary trees
   • representation
   • traversal
Priority queues
  • Linear lists
  • Heaps
Search trees
  • Binary search trees
  • balanced binary search trees – AVL trees
Graphs
  • representation
  • traversal

• ADT specification
  • stack – LIFO
  • queue – FIFO
  • (dequeue)
• implementation
  • representation using arrays
    • “circular” for queues
  • representation using linked lists
  • $\Theta(1)$ time operations
  • min and nextMin 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

Hash tables

Binary trees
  • representation
  • traversal

Priority queues
  • Linear lists
  • Heaps

Search trees
  • Binary search trees
  • balanced binary search trees - AVL trees

Graphs
  • representation
  • traversal

• 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**

<table>
<thead>
<tr>
<th>Stacks and Queues</th>
<th>Hash tables</th>
<th>Binary trees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• representation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• traversal</td>
</tr>
</tbody>
</table>

**Priority queues**
- Linear lists
- Heaps

**Search trees**
- Binary search trees
- balanced binary search trees

**Graphs**
- representation
- traversal

---

<table>
<thead>
<tr>
<th>Priority queues</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADT specification</td>
<td></td>
</tr>
<tr>
<td>• create/ destroy/ isEmpty</td>
<td></td>
</tr>
<tr>
<td>• insert</td>
<td></td>
</tr>
<tr>
<td>• min</td>
<td></td>
</tr>
<tr>
<td>• deleteMin</td>
<td></td>
</tr>
<tr>
<td>Linear lists</td>
<td></td>
</tr>
<tr>
<td>Binary tree</td>
<td></td>
</tr>
<tr>
<td>• represented using array</td>
<td></td>
</tr>
<tr>
<td>• O(log n) operations</td>
<td></td>
</tr>
<tr>
<td>• fast implementations (bit-manipulation)</td>
<td></td>
</tr>
<tr>
<td>Other operations</td>
<td></td>
</tr>
<tr>
<td>• max</td>
<td></td>
</tr>
<tr>
<td>• decrease/ increase</td>
<td></td>
</tr>
<tr>
<td>• delete</td>
<td></td>
</tr>
</tbody>
</table>
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
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees - AVL trees
- Graphs
  - representation
  - traversal

- 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)
Data representation and lists
  Hash tables
  Binary trees
    • representation
    • traversal
Priority queues
  Linear lists
  Heaps
Search trees
  Binary search trees
  • balanced binary search trees
Graphs
  • representation
  • traversal

• dynamic dictionaries -- ADT
  • create/ destroy
  • insert
  • delete
  • find

• implementation: representation using arrays as tables
  • a hash function maps keys to buckets
  • collisions may result in overflow
  • handling overflows:
    • open addressing
      • linear probing
      • quadratic probing
    • chaining
  • performance: worst-case \(O(n)\), average-case \(O(1)\)
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
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees - AVL trees
- Graphs
  - representation
  - traversal

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