CMPT 225

Data Structures and Programming

Course Website

- http://www.cs.sfu.ca/CC/225/johnwill/

Assessment

- Assignments and labs – 30%
- Midterm exam in class – 20%
- Final exam – 50%

CMPT 225 Topics

CMPT 225 Topics

- Data Structures
- Algorithms
- Software Development
- Programming

CMPT 225 Topics

- Data Structures and Abstract Data Types
  - Stacks
  - Queues
  - Priority Queues
  - Trees
  - Hash Tables
  - Graphs (if time)
- Algorithms
- Software Development
- Programming

CMPT 225 Topics

- Data Structures
- Algorithms
  - Tools – Recursion
  - Efficiency – O Notation
  - Algorithms to support data structures
  - Sorting
- Software Development
- Programming

CMPT 225 Topics

- Data Structures
- Algorithms
- Software Development
  - Specification
  - Design – OOP design
  - Implementation
  - Testing
- Programming

CMPT 225 Topics

- Data Structures
- Algorithms
- Software Development
- Programming
  - Implementing data structures and algorithms
  - Understanding stack and heap memory use
  - Recursion
  - Writing robust re-usable programs

Objectives

Objectives – Data Structures

- At the end of this course you should, for each of the data structures we cover, be able to
  - Describe the operations
  - Explain common implementations
  - Implement in a programming language (C++)
  - Compare with other data structures
  - Recommend which data structure to use for a given problem

Objectives – Algorithms

- At the end of this course you should, for each of the algorithms we cover, be able to
  - Implement in a programming language (C++)
  - Analyze running time and space requirements
  - Compare with other algorithms of a similar nature

Objectives – Programming

- At the end of this course you should be able to
  - Write algorithms using recursion
  - Understand the advantages of disadvantages of using recursive algorithms
  - Implement data structures using both arrays and reference structures as the underlying structure
  - Compare array and reference structure implementations
  - Use features of the C++ language to write well-structured programs

Objectives – O Notation

- At the end of this course you should be able to
  - Understand and describe the mathematical basis of O notation
  - Compute the O notation running time of algorithms
  - Understand the limitations of O notation

Overall Objectives

- Develop problem solving techniques
  - To take a problem statement
  - And develop a computer program to solve the problem
- A solution consists of two components
  - Algorithms
  - Data storage

Course Focus

- Problem solving
  - Use abstraction to design solutions
  - Design modular programs
  - Use recursion as a problem-solving strategy
- Provide tools for the management of data
  - Identify abstract data types (ADTs)
  - Examine applications that use the ADTs
  - Construct implementations of the ADTs

What Makes a Good Solution?

- A good solution is cost effective
  - We should minimize the cost of the software
- Running costs
  - Resources (computing time and memory)
  - Interaction costs (e.g. poor GUI may result in the loss of business)
  - Costs related to errors (e.g. loss of customer information, storing incorrect data, etc.)
- Development and maintenance costs
  - i.e. costs related to the software life cycle

Good Software Is

- Well structured
  - Modular
  - Modifiable
  - Written with good style
- Well documented
- Easy to use
- Efficient
- Able to degrade gracefully (fail-safe)
- Debugged