CMPT 225

Lecture 3 – Data collection List ADT
If we are new to OO and C++ ...

... have a look at
- Appendices A, K, L and C++ Interlude 1 (C1.1 and C1.2) of our textbook
- Any online resources such as C++ tutorials
- And remember:
  - if you find helpful web pages out there, send me their links and I will add them to our Resources page on our course web site
Academic Dishonesty

What is cheating?
- Sharing code: by copying, retyping, looking at, or supplying a file
- Describing: verbal description of code from one person to another
- Tutoring: helping your friend or you being helped by someone else to write a program
- Searching the web for solutions
- Copying code from a previous course or online solution
  - You are only allowed to use code supplied by the instructor

What is NOT cheating?
- Explaining how to use systems or tools
- Helping others with high-level design issues

See the course outline for details (in Registrar Notes section)
- Ignorance is not an excuse
Academic Dishonesty - Consequences

- Penalty for cheating:
  - Get 0 for assignment/examination
  - Permanent letter in your record

- Don’t do it!
  - Start early
  - Ask TA’s and instructor for help when you get stuck

- Academic Integrity: your work, your success
Last Lecture

- Abstract Data Type (ADT)
  - Definition + “Wall” metaphor
  - How to design an ADT
  - How to implement an ADT in C++
  - How to test an ADT
- Example: Temperature class
  - Implemented as an ADT
  - Implemented as a non-ADT
- Compare both implementations -> Homework
Homework

Advantages and disadvantages of ADT

- Advantages:
  - Preserve the integrity (invariant) of a class

- Disadvantages:
  - Must have getters/setters methods
    - so more code -> more testing
Learning Outcomes

At the end of this lecture, a student will be able to:

- define *data collection* (designed and implemented as an abstract data type - ADT) and *data structure* (concrete data type - CDT) and differentiate between the two
- define one of the concrete data types, namely *array*, and demonstrate, simulate, and trace its operations
- convert specifications into high-level design, apply software engineering and design concepts, and express OO design in UML class diagrams
- write C++ code
- encapsulate methods and variables for an ADT into a C++ class
Today’s menu

- Introduce our first data collection: List
- Design List as an ADT
  - What is hidden behind the wall
  - And what is not -> List’s public interface
- Implement List as an ADT
  - Array-based implementation
Data Collection versus Data Structure
(abstract data type - ADT) (concrete data type - CDT)

- ADT that models a collection of data
- This collection of data is what is hidden behind the wall
- Example: List

- Data structures are constructs available as part of a programming language
- Examples:
  - array
  - linked list (nodes and pointers)
Categories of data organizations

- **Linear**
  - Data organization in which each element has a unique predecessor (except for the first element, which has none) and a unique successor (except for the last element, which has none)

- **Non-Linear**
  - Data organization in which there is no first element, no last element and for each element, there is no concept of a predecessor and a successor
Categories of data organizations – cont’d

- **Hierarchical**
  - Data organization in which each element has only one predecessor -> its parent (except for the first element, which has none) and up to many successors (except for the last element(s), which has none)

- **Graph**
  - Data organization in which each element can have many predecessors and many successors
We have another problem!

- Problem Statement
- FriendsBook Application
  - Design and implement an application that maintains the data for a simple social network.
  - Each person in the network must have a profile that contains the person’s name, optional image, current status and a list of friends.
  - Your application must allow a user to join the network by creating a profile, leave the network, modify the profile, search for other profiles, and add friends.

Source: Textbook - Programming Problem 11 - Chapter 9 - Page 288
Step 2 – Design solution
Step 2 - Design List as an ADT

The wall of a data collection ADT isolates its data structure from the client code that uses it.
Note about Data Collections

- Most of the time, in a software development project, we do not design and implement data collections ADT classes, instead we make use of what is already available
  - Examples: STL vectors

- But in CMPT 225, we will design our own data collection ADT classes -> Why?

- This means that in our assignments, we cannot make use of library data collections ADT classes (like STL vectors)
Two kinds of List

- **Position-oriented**
  - Operations done based on position of data

- **Value-oriented**
  - Operations done based on value of data
Activity: Step 2 – Design
List’s public interface
Learning Check

- We can now ...
  - Explain the difference between a data collection (designed and implemented as an abstract data type - ADT) and data structure (concrete data type - CDT) and differentiate between the two
  - List different categories of data organization
  - Describe a list ADT
  - Design a list ADT
    - Design its public interface (public section)
Next Lecture

- Finish the design of a list ADT
  - Design its underlying data structure (part of its private section)
- Implement the array-based List ADT and test it
- Introduce test cases